VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

IDENTIFIKACE APLIKAČNÍCH PROTOKOLŮ

APPLICATION PROTOCOLS IDENTIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ CHOMO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN PLUSKAL,**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20191/2016/xchomo00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Chomo Tomáš**
Obor: Informační technologie
Téma: **Identifikace aplikačních protokolů**
Application Protocols Identification
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s problematikou identifikace aplikačních protokolů z doporučené literatury.
2. Seznamte se s prostředím Netfox Detective a stávajícím modulem pro rozpoznání aplikačních protokolů.
3. Vytvořte si testovací dataset, kterým ověříte aktuální přesnost identifikačního modulu.
4. Navrhněte vhodná rozšíření a implementujte je.
5. Otestujte navržená rozšíření a vyhodnoťte, jak se identifikace zpřesnila oproti stávající implementaci.

Literatura:

- Foroushani, V. A., & Zincir-Heywood, A. N. (2013). Investigating application behavior in network traffic traces. In *Proceedings of the 2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013* (pp. 72-79).
- Korczyński, M., & Duda, A. (2014). Markov chain fingerprinting to classify encrypted traffic. In *Proceedings - IEEE INFOCOM* (pp. 781-789). Institute of Electrical and Electronics Engineers Inc.
- Miskovic, S., Lee, G. M., Liao, Y., & Baldi, M. (2015). AppPrint: Automatic fingerprinting of mobile applications in network traffic. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 8995, pp. 57-69). Springer Verlag.
- Hajjar, A., Khalife, J., & Díaz-Verdejo, J. (2015). Network traffic application identification based on message size analysis. *Journal of Network and Computer Applications, 58*, 130-143. Retrieved from http://www.sciencedirect.com/science/article/pii/S1084804515002167

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Pluskal Jan, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Digitálna forenzná analýza aplikuje metodické sady techník a procedúr potrebných na získanie dôkazov z počítačových zariadení a prezentuje v zmysluplnom formáte. Táto práca sa zaoberá problematikou identifikácie aplikačných protokolov za pomoci metód strojového učenia a štatistických metód. V práci je obsiahnuté testovania aktuálnej implementácie agenta Netfox Detective, ktorý využíva tieto dve metódy. Následne som sa snažil zlepšiť detekčné schopnosti pomocou procesu zvaného Feature Engineering, ktorého úlohou je vytvárať sadu príznakov, ktoré nám môžu pomôcť charakterizovať sieťovú komunikáciu. Práca porovnáva tieto dve metódy detekcie a rozširuje ich implementáciu s úsilím vylepšiť detekčné schopností agenta Netfox Detective.

Abstract

Digital forensic analysis applies methodical series of techniques and procedures used to gather evidence, from computer device and present it in meaningful format. This thesis is dealing with identification of aplication protocols with help of machine learning and statistical methods. Further thesis explain attempts to improve detection skills with help of process called Feature Engineering. Feature Engineering is process of creating set of features that will help us to characterise network traffic. Paper contains testing of actual implementation of agent Netfox Detective which uses those two methodes Paper is comparing those two methods and extends the implementation with effort to improve detection skills of a Netfox Detective agent.

Kľúčové slová

identifikácia, klasifikácia sieťovej komunikácie, aplikačný protokol, strojové učenie, analýza, digitálna forenzná analýza

Keywords

identification, network communication classification, application protocol, machine learning, digital forensic analysis

Citácia

CHOMO, Tomáš. *Identifikace aplikačních protokolů*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal,

Identifikace aplikačních protokolů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Jana Pluskala. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Tomáš Chomo
17. mája 2017

Poďakovanie

Týmto by som chcel poďakovať vedúcemu práce, Ing. Janovi Pluskalovi, za ochotu a čas, ktorý mi venoval pri vypracovaní tejto práce.

Obsah

Kapitola 1

Úvod

Vo svete ovládaného informačnými technológiami, je väčšina našej komunikácie práve na internete. Táto komunikácia je prenášaná po sieti ako tok dát, ktorý v sebe nesie všetky potrebné informácie aby nám toto spojenie zabezpečil. Schopnosť porozumieť tomuto toku dát je v dnešnej dobe veľmi žiadaná. Z dát je možné rôznymi spôsobmi vyčítať informácie, ktoré nám môžu popísať chovanie užívateľov na sieti. Môžeme zistiť napríklad akým spôsobom sa pripájal ku internetu, ako dlho a s kým komunikoval. Podrobnejšou analýzou môžeme, taktiež zistiť aké programy na svojom zariadení používal. Táto znalosť má využitie vo viacerých odvetviach informačného zamerania. Hlavným využitím sa aktuálne stáva detekcia a prevencia potencionálne nežiadanej sieťovej premávky, teda systém detekcie narušenia (IDS)[1]. Detailné informácie o aplikačných protokoloch na sieti môžu taktiež významne dopomôcť ku zaistení kvality služieb (QoS)[2]. Táto práca sa zameriava sieťovou forenznou analýzou za pomoci strojového učenia a štatistických metód, ktorá je odvetvím digitálnej forenznej analýzy.

Strojové učenie dovoľuje zariadeniu učiť sa bez potreby explicitného programovania. Pomocou tejto metódy sa algoritmus môže z pred pripravených dát naučiť črty sieťovej premávky a nesladne predpovedať o akú typ komunikácie ide. V posledný rokoch sa dostáva metóda strojového učenia významne do popredia, pretože zovšeobecňuje a zlepšuje doterajšie implementácie algoritmov.[27] Aplikícia NetFox Detective[3] využíva štatistické metody a metódy strojového učenia na identifikáciu aplikačných protokolov v sieťovej prevádzke. Tento produkt je vyvíjaný na Fakultě informačních technologií Vysokého učení technického v Brně.

Cieľom tejto práce je zlepšenie identifikáciu aplikačných protokolov aplikácie Netfox Detective. Zlepšenie identifikácie zahŕňa rozšírenie a otestovanie príznakov, ktoré slúžia na štatistický popis tokov v sieťovej premávke. Následne je potrebné správne neimplementovať a odladiť váhy, ktoré budú týmto metrikám priradené. Vypočítané metriky sú využité klasifikátorom, ktorý slúži na detekciu aplikačných protokolov pomocou metódy strojového učenia a štatistickej metódy.

V nasledujúcom texte je prezentovaný rozbor rozličných prístup ku identifikácií aplikačných protokolov popísaný v kapitole 2. O prehľade metód strojového učenia používaných pri klasifikácií dát hovorí kapitola 3. V krátkosti popísaná problematika výkonnosti a porovnanie úspešnosti rôznych princípov. V kapitole 4 je spomenuté odvetvie Feature Engineeringu, ktorého poznatky boli použité pri pokuse o zlepšenie identifikačných schopností. Kapitola

[1] Intrusion detection system https://en.wikipedia.org/wiki/Intrusion_detection_system
[2] Quality of service https://en.wikipedia.org/wiki/Quality_of_service
[3] NFX Detective http://netfox.fit.vutbr.cz/About.en.html

číslo 5 hovorí o samotnom priebehu implementácie a následnom testovaní rozšírení aplikácie Netfox Detective. V závere tejto práce sú zhrnuté výsledky testovania a implementácie, problémy pri implementácií a následné návrhy na dodatočné vylepšenie implementácie. Príloha A popisuje rôzne rôzne metódy strojového učenia. Priloha B obsahuje tabuľky hodnôt testovaní.

Kapitola 2

Varianty identifikácie aplikačných protokolov

Táto kapitola rozoberá dnes používané metódy a prístupy ku identifikácií aplikačných protokolov. Uvádza problémy, ktoré môžu nastať pri špecifických metódach. Ďalej popisuje výhody a dôvody na zvolenie daných metód identifikácie. Medzi tri základné metódy slúžiace na identifikáciu aplikačných protokolov patrí *identifikácia na základe čísla cieľového portu*, *analýza payloadu* a *identifikácia na základe štatistík tokov* [14][13].

Pri klasifikácií aplikačných protokolov je potrebné vedieť ako sa daný protokol správa a aké sú jeho charakteristike vlastnosti v sieti. Tieto informácie môžeme získať analýzou sieťovej premávky, rozborom aplikačnej vrstvy[3][18] a taktiež sa tieto informácie nachádzajú v dokumentáciách daných protokolov. Klasifikovať toky môžeme manuálne za pomoci programov ako je Wireshark [1], alebo automaticky.

Manuálna analýza sieťovej prevádzky je značne zdĺhavá a neefektívna[17]. Forenzné rozbory tokov sa vykonávajú v laboratórnych prostrediach ich dĺžka závisí od veľkosti dát potrebných ku analýze a počte ľudí pracujúcich na tomto probléme[17]. V týchto prípadoch je automatizácia tohoto procesu viac než vítaná.

Automatické metódy rozboru nám umožňujú rýchlo a dôsledne identifikovať charakteristiky sieťového toku za pomoci minimálnych požiadavok na personál. Umožňuje nám prístup ku aktuálnemu prehľadu informácií. Metódy používané pri automatizácií identifikácie aplikačných protokolov sú popísané v tejto kapitole. Pri automatickej analýze sa môžeme zamerať na algoritmy hľadajúcej špeciálne reťazce vyskytujúce sa v aplikačných dátach (napr. SNORT [2],l7-filter[3]). Určité algoritmy si všímajú špecifické chovanie protokolov[15], alebo zbierajú štatistiky nad dátami[30]. Tieto štatistiky sú následne použité pri klasifikácií protokolov.

Jedným z problémov, na ktoré určite narazíme pri získavaní inšpekcií sieťovej prevádzky je šifrovanie dát z dôvodu bezpečnosti. V dobe, kedy sa veľký dôraz kladie na bezpečnosť internetovej komunikácie, môže byť veľmi obtiažne zistiť o aký druh protokolu sa jedná. Pred príchodom masového šifrovania komunikácie sa dali jednoduchým zachytávaním dát získať citlivé informácie, ktoré boli mohli byť obsiahnuté v komunikácií. Vďaka tomuto bolo jednoduchšie presnejšie klasifikovať sieťovú premávku. Aj napriek tomu že šifrované

[1] Wireshark je analyzér sieťových protokolov https://www.wireshark.org/
[2] Snort - Network Intrusion Detection & Prevention System https://www.snort.org/
[3] l7-filter http://l7-filter.sourceforge.net/

dáta potrebujú kľúč, ku ktorému dosť často nemáme prístup, je možné ich analýzou získať niektoré charakteristiky pomocou štatistického prístupu ku analýze.

2.1 Identifikácia pomocou portov

Jednou z najpoužívanejších techník identifikácie aplikačných protokolov je technika identifikácie na základe čísla cieľového portu. Táto metóda je jednoduchá na implementáciu a nieje výpočetne náročná. Bohužiaľ má táto metóda aj značné nedostatky akými sú veľmi vysoká nepresnosť [22].

Na transportnej vrstve adresujeme služby bežiace na zariadeniach. Adresu služby prenáša vo svojej hlavičke protokol TCP alebo UDP vo forme 16-bitového čísla portu. Číslo portu identifikuje službu, ktorá dáta posiela (číslo portu odosielateľa), alebo cieľový aplikačný proces (číslo portu adresáta). Číslo portu adresáta musí byť obecne známe. Napr. pri komunikácií s webovým klientom sa dotazujeme na port číslo 80, alebo pri prenose súborov cez ftp na port 21. Takýmto službám sa hovorí well-known-services. Je možné mapovanie čísla portov na iné porty. Bežne sa využíva mapovania portu 80 na port 8080. Keby sme nevedeli číslo portu služby s ktorou chceme komunikovať nedokázali by sme ju využívať. Naopak čísla portov odosielateľov sú náhodne generované pri vytvorení komunikácie.

Čísla portov sa delia na rezervované (0–1023), registrované (1024–49151) a dynamické (49452–65535). Rezervované porty prideľuje organizácia IANA [4] (Internet Assigned Numbers Authority) pre štandardné služby, napríklad FTP, SNMP, HTTP apod. Bohužiaľ existuje mnoho aplikácií, ktoré nemajú tieto porty pridelené a to dosť často preto lebo používajú rôzne čísla portu pre svojej pripojenie. Napríklad aplikácie pre streamovanie videí, alebo sťahovanie torrentov. Taktiež je táto detekcia neúspešná ak je jedna strana komunikujúcej dvojice za NAT zariadením.

Práve čísla well-known-services sú tie, ktoré nás zaujímajú pri identifikácií. Identifikácia prebieha na základe čísla portu získaného na na transportnej vrstve z protokolu TCP a UDP.

Práve pre príčiny, ako sú mapovanie portov na iné porty a využívanie dynamických portov je identifikácia používajúca metódu známych portov veľmi nepresná. Jej úspešnosť dosahuje iba okolo 50% až 70%.[23] Tieto čísla sú pre dnešné účely veľmi malé a nepostačujúce [15].

2.2 Analýza payloadu

Metódu bližšie popísanú v tejto časti textu aktuálne využíva väčšina mechanizmov na identifikáciu. Táto metóda je omnoho presnejšia ako identifikácia za pomoci čísla portov, pretože využíva viacej dát podľa, ktorých určuje charakteristiky odpovedajúce určitým protokolom. Metoda analýzy payloadu[5], teda analýza dát prenášaných v rámci aplikačnej vrstvy TCP/IP. Pri analýze dát sa zameriavame na opakujúce sa podreťazce, ktoré sú špecifické pre daný aplikačný protokol. Napríklad protokol HTTP obsahuje kľúčové reťazce "HTTP", protokol SIP môže obsahovať "Via", "From", "To", alebo protokol IMAP obsahuje podreťazce ako "Subscribe". Tieto špecifikácie môžeme nazvať aj odtlačkom protokolov[6]. Analýzou a identifikáciou vlastností protokolu dokážeme určiť jeho charakteristiku a veľmi spoľahlivo identifikovať protokol[25]. Nevýhodou, s ktorou sa stretávame pri tejto variante sú šifrované

[4] Internet Assigned Numbers Authority http://www.iana.net/

[5] Payload je časť prenášaných dát, ktorá obsahuje určenú správu

[6] Odtlačok protokolu je charakteristika, ktorá identifikuje protokol.

dáta. Pri šifrovanom toku dát sa nemôžeme dostať ku detekcií podreťazcov čo nám úplne znemožní detekciu. Taktiež pri analýze payloadu na reálnej premávke narážame na vysoké výkonnostné požiadavky detektoru. Preto systém s touto funkcionalitou musí byť veľmi precízne optimalizovaný a udržiavaný. Tento princíp sa využíva v IDS (intrusion detection system) systémoch. IDS [24] je software, ktorý monitoruje sieť alebo systém a všíma si podozrivej aktivity. Mechanizmus detekuje nežiadúce správanie na sieti špecifikované pravidlami IDS. Vitálnou zložkou detekcie je práve analýza payloadu, ktorá poskytuje dostatočné informácie, ktoré porovnáva s pravidlami. V prípade zhody je incident oznámený užívateľovi.

Existuje mnoho druhov algoritmov na identifikáciu pomocou analýzu dát, ktoré paket obsahuje. Napríklad algoritmus AutoSig [5], ktorý slúži na získanie podpisu aplikácie. Tieto podpisy sú následne uložene do stromovej štruktúry, ktorá uľahčuje prácu pri identifikácií. Ďalším z používaných metód je klasifikátor L7-filter. Ten identifikuje aplikačný protokol na základe aplikačnej vrstvy. Využíva k tomu databázu podpisov aplikačných protokolov. Položky tejto databáze majú tvar regulárnych výrazov, ktoré dokážu určiť aplikačných protokol. Táto databáza je ale tvorená a udržiavaná ručne pomocou manuálne inšpekcie paketov. Rozširovanie podpisov je náročné a zdĺhavé. Jedným z prepracovaných prístupov ku tejto problematike rozobrali vo svojej práci Andrew W. Moore a Konstantina Papagiannaki [15]. Využívajú iteratívny prístup ku klasifikácií tokov a snažia sa pomocou viacerých krokov určiť aplikačný protokol komunikácie. Tento prístup zahrňuje viacero techník identifikácie, ktorými daný tok musí prejsť a v rámci týchto krokov bude vyhodnotení. Klasifikácia zahŕňa techniky detekcie portov, inšpekcia hlavičky paketov a vylučovanie tokov obsahujúcich malé množstvo paketov. Podpisovanie jedného paketu, protokol jedného paketu, podpis na prvý kilobajt princípe atď. Algoritmus postupne aplikuje všetky pravidlá na tok a pokiaľ daný test jednoznačne určí protokol, tak sú ostatné prístupy preskočené. Vďaka tomu že metóda pracuje nad tokmi je schopná odhaliť nežiadúce správania v premávke a iné nebezpečenstvá ako sú počítačové vírusy. Nevýhodou tohoto prístupu je nutnosť spolupráce s človekom, ktorý musí manuálne upravovať výsledok identifikácie ak sa korektne nepodaril. Ďalšou s nevýhod je už vyššie spomínaná potreba nešifrovaného dátového toku.

2.3 Identifikácia na základe štatistik získaných zo sieťovej komunikácie

Identifikácia na základe flow[7] štatistík, využíva dáta získané z hlavičiek IP a TCP protokolu [14] [30]. Keďže šifrovaná je až aplikačná vrstva, táto technika si poradí aj so šifrovanou komunikáciou Štatistiky sa počítajú pre jednu konverzáciu (obojsmerný tok medzi dvoma koncovými bodmi) a berú sa do úvahy oba smery osobitne. Takto nám vznikajú štatistiky v smere klient – server a aj smere server – klient. Jedným z bežne používaných štatistik metód sú [30] [14]:

- celková veľkosť zaslaných paketov,
- počet zaslaných paketov,
- doba trvania konverzácie,
- minimálna, priemerná a maximálna veľkosť paketov,

[7] Flow je sekvencia paketov, ktorá ide zo zdrojového zariadenia do koncového.

- minimálna, priemerná a maximálna doba medzi dvoma paketmi,
- odchýlka veľkosti paketov,
- jitter[8].

Tieto príznaky následne tvoria charakteristiku vlastností pre daný protokol.

2.4 SPID - Statistical Protocol IDentification

Tento algoritmus je schopný identifikácie aplikačných protokolov. Jeho autorom je Erik Hjemvikem[7]. Táto metóda kombinuje vlastnosti analýzy payloadu a identifikácie na základe flow štatistík. Jedným z hlavným cieľom SPIDu sú malé databáze známych aplikačných protokolov, nízka časová náročnosť, identifikácia na malom vzorku dát, spoľahlivosť a presnosť. Vďaka týmto cieľom je využiteľný na klasifikáciu premávky v reálnom čase. Tento algoritmus využíva veľké množstvo metrických a funkcií, ktoré využíva na získavanie potrebných informácií na detekciu charakteristík protokolu a vytvára si vlastná databázu odtlačkov aplikácií. Tento algoritmus sa radí medzi algoritmy s učiteľom (ďalej popísané v sekcií A.2) a pred prvým spustením vyžaduje predpripravenú databázu, alebo predom pripravené klasifikované dáta pre vytvorenie takej databáze.

SPID pracuje iba s konverziami nad TCP a zvládne identifikovať aplikačný protokol na prvých štyroch aplikačných paketoch, maximálne však využije sto paketov. Do samotnej klasifikácie teda nezahŕňame kontrolne a signalizačné pakety. Otlačok protokolu má podobu dvoch vektorov. Vektor vlastností a pravdepodobnostný vektor. Vektor vlastností obsahuje výhradne nezáporné čísla a jeho položky upravuje algoritmus pomocou 34 metrických funkcií. Medzi najužitočnejšie funkcie patria:

- akcia reakcia prvých 3 bajtov - detekcia odpovede serveru k dotazu,
- frekvencia bajtov - rozloženie bajtov v správe,
- počet opakujúcich sa bajtových párov v prvých 32 bajtoch - počet opakujúcich sa dvojíc bytov,
- prvé 2 usporiadané 4 znakové slová - detekcia protokolov s fixnými hlavičkami,
- pozícia prvého bitu- záznam pozície bitov s hodnotou 1.

Každá z takýchto metrických funkcií má svoju dvojicu vektorov a ovplyvňuje rôzne položky vektoru vlastností. Môžeme povedať že vektor vlastností obsahuje počet jednotlivých výskytov atribútov. Normalizovanou verziu vektoru vlastností predstavuje pravdepodobnostný vektor. Jeho položky môžu nadobúdať hodnoty v rozsahu 0.0 až 1.0 a súčet v rámci vektoru je vždy rovný 1. Implementáciu SPIDu umožňuje ľubovolnú veľkosť vektorov, ale východzia hodnota je vždy nastavená na 256. Získaný odtlačok protokolu je tvorený pravdepodobnostnými vektormi všetkých metrických funkcií. Model protokolu sa skladá zo všetkých odtlačkov daného protokolu a je získaný buď zo predpripravenej databáze ale už z klasifikovaných dát.

$$DKL(P_{atr}||Q_{protoAtr}) = \sum_{i}(P_{atr} * \log \frac{P_{atr}(i)}{Q_{protoAtr}(i)}) \tag{2.1}$$

[8] Jitter je odchýlka od periodicity od periodického signálu.

Samotné porovnanie prebieha pomocou Kullback-Leiblerovej divergencie (viac rovnica 2.1), kde P_{atr} predstavuje pravdepodobnostný vektor špecifickej metrickej funkcie sledovanej konverzácie a $Q_{protoAtr}$ predstavuje hodnotu získanú z modelu protokolu. Konverzácií je potom priradený model protokolu s najmenšou hodnotou Kullback-Leiblerovej divergencie.

Kapitola 3

Klasifikácia dát pomocou metód strojového učenia

Štatistická analýza a identifikácia aplikačných protokolov potrebuje určité množstvo dát na vytvorenie charakteristík, ktoré by boli spoľahlivé. Tým potrebný na manuálnu analýzu niekoľko dňovej a možno až týždňovej sieťovej prevádzky by bol veľmi drahý a časovo náročný[17]. Využívanie manuálnych techník na vytváranie štatistik je náchylné na chybovosť[11]. Presne pri takýchto prípadoch nám môžu prácu značne uľahčiť techniky strojového učenia v oblasti analýzy a identifikácie aplikačných protokolov, ktoré sú detailnejšie rozobrané v tejto kapitole. Pomocou týchto metód dokážeme vytvoriť model, ktorý následne využijeme pri klasifikácií dát[13]. Metódy strojového učenia sú známe svojou schopnosťou učiť sa. Následne sa delia prístupom ku ich učiacej fáze. Učenie s učiteľom, učenie bez učiteľa a kombinácia týchto dvoch.

3.1 Klasifikácia dát

V oblasti strojového učenia a štatistiky je hlavnou úlohou klasifikácie dát identifikovať, ku ktorému súboru kategórií patrí nové pozorovanie, na základe trénovacej sady dát obsahujúcej pozorovania, ktorých kategorizácia je známa. Teda vytvorenie charakteristík aplikačných protokolov. Toto je úlohou prvej časti tejto metódy a to učiacej časti. Následne sa vytvorí z daných dát klasifikátor, ktorý pomáha klasifikovať ostatné dáta[1][21].

Na vytvorenie klasifikátora potrebujeme cvičné dáta, ktoré pomôžu vytvoriť charakteristiky[23]. Trénovacia množina dát sa skladá z dôležitých a sledovaných vlastností pre konkrétny prípad a k ním priradením triedam. Charakteristické vlastnosti sú N-ticou s hodnotou, ktorá predstavuje výslednú triedu. V prvom kroku klasifikácie teda dochádza ku vytvoreniu klasifikačných pravidiel. V tejto fáze je cieľom vytvoriť sadu funkcií, príznakov, ktoré od seba dostatočne odlíšia priradené triedy[2][21]. V ďalšom kroku dochádza ku testovaniu novo vzniknutého modelu na množine testovacích dát, ktorá je nezávislá na učiacej množine. Trochu odlišný prístup ku učeniu ponúka metóda učenia bez učiteľa, ktorá pre svoju činnosť nepotrebuje učiacu množinu s predom známymi triedami. Tieto metódy, medzi ktorých zástupcov patrí napríklad K-means [9] alebo K-memoids[10], pracujú na princípe zhlukovej analýzy a môžu nám pomôcť odhaliť novu triedu[11]. Keď spojíme oba vyššie zmienené princípy, vznikne nám princíp kombinovaného učenia, ktoré v sebe zahrňuje ako učenie s učiteľom,

[1] Klasifikátor je algoritmus, ktorý implementuje klasifikáciu dát. Mapuje vstupné dáta na kategórie.

[2] Iris flower data set "https://en.wikipedia.org/wiki/Iris_flower_data_set"

teda aj schopnosť vytvoriť model na základe už klasifikovaných dát, tak aj učenie bez učiteľa a jeho schopnosť identifikovať novú triedu.

3.2 Učenie s učiteľom - Supervised learning

Obecne nám môžu metódy učenia s učiteľom pomôcť pri klasifikácií, kedy máme predom danú množinu s už klasifikovanými dátami. Medzi metódy učenia s učiteľom patrí Bayesovská klasifikácia[16](sekcia 3.2.1), siete(sekcia 3.2.1), rozhodovanie stromy(sekcia A.1.1), algoritmus k-najbližších susedou (sekcia A.1.3) alebo neurónové siete(sekcia A.1.2).

3.2.1 Bayesová klasifikácia

Bayesová klasifikácia je štatistická metóda založená na Bayesovom teoréme podmienených pravdepodobností (rovnica 3.1), ktorý udáva pravdepodobnosť, že platí hypotéza H, ak sledujeme X.

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \tag{3.1}$$

Pričom $P(H)$ predstavuje apriórnu pravdepodobnosť hypotézy H, teda zastúpenie jednotlivých hypotéz. $P(H)$ je pravdepodobnosť pozorovania javu X a $P(H|X)$ hovorí o pravdepodobnosti dát X, ak je hypotéza H pravdivá. Pomocou Bayerovského teorému hladme jav s najväčšou pravdepodobnosťou výskytu na základe vstupných dát. Taký jav nazývame jav s maximálnou posteriórnou pravdepodobnosťou a získame ho pomocou vzťahu 3.2 [11][16]. Vzhľadom k tomu, že hodnota $P(X)$ je pre všetky H_i rovnaká, môžeme ju ignorovať.

$$P_{MPP} = maxP(X|H_i)P(H_i) \tag{3.2}$$

Teorém popísaný v rovnici 3.1 a maximálna posteriorna pravdepodobnosť v rovnici 3.2 sa ďalej využívajú v naivnej Bayerovskej klasifikácií a Bayesovských sieťach.

Naivná Bayesovská klasifikácia

Naivná Bayesovská klasifikácia je jednoduchá na implementáciu a dosahuje veľmi malej chybovosti v porovnaní z ostatnými klasifikátormi pracujúcimi na princípe učenia s učiteľom. Pre vytvorenie modelu nepotrebuje veľké množstvo dát[11]. Agent Netfox Detective ju využíva ako metódu strojového učenia. Vychádza z predpokladu nezávislosti atribútov. Efekt hodnoty každého atribútu na danú triedu, nieje ovplyvnený hodnotami ostatných atribútov. Pre nespojité atribúty, je to súčin všetkých podmienených pravdepodobností pre jednotlivé hodnoty atribútov. Nezávislosťou je možné znížiť výpočetnú náročnosť pri veľkom počte, parametrov, tedy môžeme použiť vzťah 3.3[11].

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \cdots \times P(x_n \mid C_i) \tag{3.3}$$

Tento vzťah je možné použiť iba v prípade, ak neexistuje žiadna podmienená väzba medzi atribútami. Pri výpočtu pravdepodobnosti musíme rozlíšiť, či sa jedná o diskrétne, alebo spojité dáta.

- Ak sa jedná o diskrétne dáta, tak pravdepodobnosť budeme počítať ako podiel počtu výskytov danej hodnoty x_k sledovanej vlastnosti v triede C_i a po4tu N-tic označených

triedov C_i.

$$P(x_k \mid C_i) = \frac{|x_k|}{|C_i|} \tag{3.4}$$

- Ak počítame so spojitými dátami, musíme použiť Gaussovské rozloženie, podľa vzorca 3.5, kde μ predstavuje strednú hodnotu a σ rozptyl.

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.5}$$

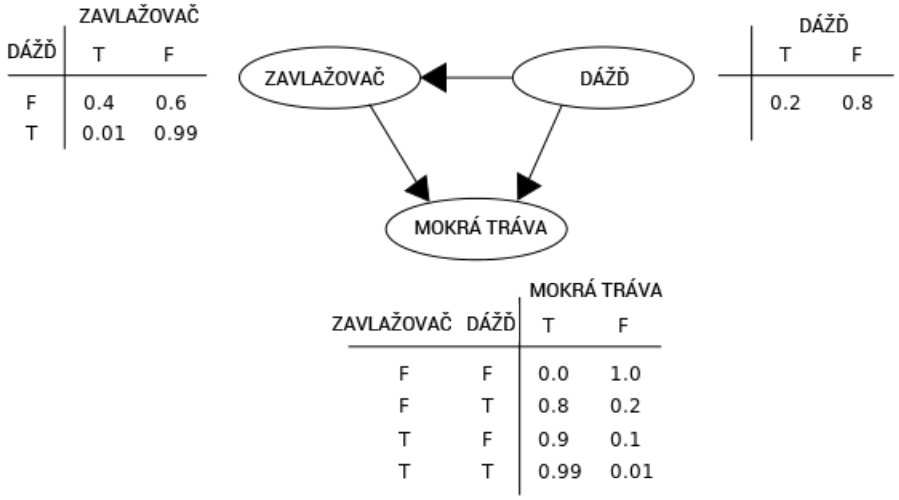
Potom môžeme pravdepodobnosť spočítať pomocou vzorca 3.6.

$$P(x_k \mid C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) \tag{3.6}$$

Pri následnej klasifikácií pre N-ticu X musíme vypočítať hodnotu $P(X \mid C_i)P(C_i)$ pre všetky triedy C_i. Výsledná N-tica X je potom trieda C_i z najväčšími hodnotami $P(X \mid C_i)P(C_i)$.

Bayesovské siete

Jednou z hlavných schopností Bayeovských sieti je schopnosť spracovávať dáta medzi, ktorými sa objavujú podmienečné vzťahy. Predstavujú pravdepodobnostný grafický model týchto závislostí, na ktorom prebieha učenie a klasifikácia. Bayesovské siete sa skladajú z orientovaného acyklického grafu a množiny tabuliek podmienených pravdepodobnosťou[8]. Každý uzol grafu odpovedá jednej náhodnej veličine, pričom každý graf typicky obsahuje niekoľko veličín. Náhodnou veličinou rozumieme vlastnosť, či už atribút zo vstupných dát alebo novo pridané vlastnosť vyplývajúcu zo závislosti na ostatných atribútoch. Každá hrana potom predstavuje pravdepodobnostnú závislosť[11].



Obr. 3.1: Jednoduchá Bayesova sieť s tabuľkou podmienených pravdepodobností

Ak hrana ukazuje z uzlu a do uzlu B, tak je uzol a rodičom čiže predchodcom uzlu B. V Bayesovskej sieti potom pripadá jedna tabuľka podmienených pravdepodobností na jednu vlastnosť uvedenú v grafe. Tabuľka atribútu Y udáva podmienenú distribúciu $P(Y|Rodic(Y))$, kde $Rodic(Y)$ sú priamymi predchodcami Y.

Uvedieme si príklad ku obrázku 3.1[3]. Existujú dve možnosti pri ktorých môže zmoknúť tráva. Dážď alebo zavlažovač môžu spôsobiť že tráva zmokne. Taktiež predpokladajme že dážď priamo má priamy vplyv na využitie zavlažovania (ak prší tak väčšinou zavlažovanie nieje spustené) Vzorec spojitého rozdelenia pravdepodobnosti (rovnica3.7):

$$\Pr(T, Z, D) = \Pr(T|Z, D) \Pr(Z|D) \Pr(D) \tag{3.7}$$

V rovnici 3.8, kde mená premenných znamenajú:

- T = Mokrá tráva(áno/nie)3
- Z = Zapnuté zavlažovanie(áno/nie)
- D = Dážď(áno/nie)

Tento model môže odpovedať na otázku ako je napríklad: Aká je pravdepodobnosť, že prší ak je tráva mokrá pomocou použitia algoritmu podmienenej pravdepodobnosti.

$$\Pr(D = T|T = T) = \frac{\Pr(T = T, D = T)}{\Pr(T = T)} = \frac{\sum_{S \in \{T,F\}} \Pr(T = T, Z, D = T)}{\sum_{Z,D \in \{T,F\}} \Pr(T = T, Z, D)} \tag{3.8}$$

Použitím rozšírenia pre spojité rozdelenie pravdepodobnostnej funkcie $Pr(T, Z, D)$ a podmienenej pravdepodobnosti z tabuliek podmienkových pravdepodobností v rovnici 3.9 môžeme vypočítať výraz v sumách čitateľa a menovateľa.

$$\begin{aligned} \Pr(T = T, Z = T, D = T) &= \Pr(T = T|Z = T, D = T) \Pr(Z = T|D = T) \Pr(D = T) \\ &= 0.99 \times 0.01 \times 0.2 \\ &= 0.00198. \end{aligned} \tag{3.9}$$

Číselný výsledok po vyčíslení jednotlivých premenných je vyobrazený v rovnici 3.10:

$$\Pr(D = T|T = T) = \frac{0.00198_{TTT} + 0.1584_{TFT}}{0.00198_{TTT} + 0.288_{TTF} + 0.1584_{TFT} + 0.0_{TFF}} = \frac{891}{2491} \approx 35.77\%. \tag{3.10}$$

Bayovské sieťe sú výpočetne náročné, ale môžu byť použité aj pre získanie rozdelenia pravdepodobnosti pre jednotlivé triedy[28].

[3]Príklad prevzatý z https://en.wikipedia.org/wiki/Bayesian_network

Kapitola 4

Feature engineering

Feature [1].[1] engineering je neformálne odvetvie strojového učenia, ktoré je kľúčom k úspechu v aplikovanom strojovom učení. Tento proces využíva doménové znalostí o dátach na vytvorenie funkcií príznakov, ktoré sú jadrom algoritmu strojového učenia. Toto odvetvie je časovo aj vedecky náročné. Vývojár manuálne vytvára potrebné funkcie, ktoré by mali byť vecné voči kontextu a mali by byť nápomocné pri klasifikácií a následnej predpovedí výsledku. Agent Netfox Detective využíva príznaky na výpočet charakteristík, ktoré sú použité ako pri metóde strojového učenia, tak aj pri štatistickej metóde. Voľba Feature Engineeringu, ako kľúčom ku zlepšeniu detekčných schopností agenta, bola inšpirovaná faktorom že táto metóda vyžaduje do detailov pochopiť štruktúru toho čo detekujeme, teda sieťovej premávky.

4.1 Príznaky

Príznaky, ktoré sú taktiež nazývané features sú akékoľvek informácie, ktoré môžu byť prospešné pre predikciu výsledku v strojovom učení, alebo štatistických metódach. Väčšina príznakov popisuje charakter sieťovej premávky namiesto obsahu individuálnych paketov. Každá informácia o vstupných dátach, ktorá je nápomocná pri klasifikácií môže byť príznakom. Výber vhodných, informatívnych a nezávislých príznakov je kľúčovým krokom ku efektívnemu algoritmu. Nespoliehať sa na špecifické vzory, ktoré sa môžu vyskytovať v tejto komunikácií prináša lepšie výsledky pri detekcií šifrovanej premávky. Príznak môžu vyzerať nasledovne, funkcia, ktorá počíta dĺžku spojenia klienta zo serverom, počet dát prenesených za jednotku času, priemerná odozva a mnoho iných.

Prvým, krokom je podrobné naštudovanie si dát nad, ktorými sa budú príznaky počítať a vymyslenie počiatočnej sady príznakov, ktoré by mohli byť prospešné. Je potrebné zamyslieť sa nad využitím sady príznakov a či hodnoty, ktoré budú vypočítané pomocou príznakov sú využiteľné v celom spektre dát. Počiatočná sada príznakov môže byť nadbytočná a príliš veľká na to aby bola udržateľná. Jedným z počiatočných krokov aplikácií z využitím strojového učenia sa stáva selekcia podmnožiny funkcií, ktoré budú vyhovovať učeniu, vylepšiť a generalizovať jeho schopnosť interpretácie. V tejto práci sme zvolili funkcie, ktoré neboli ovplyvnené hardvérovou stránkou sieťovej komunikácie, aby sme eliminovali výkyvy, ktorú sú spôsobené výkonnosťou sieťových prvkov.

Výber vhodných funkcií je kombináciou vedy a umenia. V tejto fáze je nevyhnutné čo najpodrobnejšie testovanie a experimentovanie so selekciou príznakov. Je potrebné zistiť,

[1] Feature, alebo príznak je individuálna merateľná vlastnosť pozorovaného javu

ktoré sú v kontexte najprospešnejšie a rozšíriť implementáciu, o taký druh príznakov. Vektor vybraných príznakov je následne použitý na zozbieranie charakteristík nad vstupnými dátami. Tieto údaje sa používajú zostavenie klasifikátorov, ktoré sú následne využívané v metódach strojového učenia a štatistickej identifikácie protokolov.

4.2 Váhy

Každému príznaku je pridelená váha, ktorá hovorí o dôležitosti hodnoty príznaku. Ku výpočtu váhy funkcií môžeme pristupovať viacerými spôsobmi. Jedným zo zaužívaných spôsobov je výpočet váhy za pomoci smerodajnej odchýlky[2] lebo entropie[3]. Pri výpočte smerodajnej odchýlky použijeme priemernú hodnotu vypočítanú z výsledkov príznakov a následne vypočítame odchýlku tejto hodnoty od všetkých hodnôt danej funkcie. Táto hodnota nám môže slúžiť ako váha, ale nemusí vždy korektne fungovať v našom algoritme. Takéto váhy ne následne potrebné manuálne preveriť a kalibrovať, tak aby boli použiteľné. Váhy následne používa štatistická metóda pri pri vytvorení modelu aplikačného protokolu vektoru príznakov v trénovacej fáze programu. Následne sa váhy využívajú aj v testovacej fáze, kde sú použité vo výpočte Euklidovskej vzdialenosti [4]. Euklidovská vzdialenosť je metóda na výpočet kvantitatívnej vzdialeností dvoch veličín, ktorá hovorí a podobnosti, resp. odlišnosti dvoch objektov.

4.3 Metriky

Pri detekcií pomocou štatistickej metódy a strojového učenia je potrebné zistiť ako presné sú tieto výsledky. V poslednom kroku detekcie sa preto počítajú štatistické metriky úspešnosti. Táto sada metrík môže obsahovať metriky skutočne pozitívnych (TP - true positive), falošne pozitívnych (FP - false positive) a falošne negatívnych (FN - false negative) klasifikácií. Ďalej sú to metriky ako je presnosť (Precision = TP / (TP + FP)), ktorá hovorí o tom koľko nálezov bolo naozaj správny. Ďalšou je nález (Recall = TP / (TP + FN)), teda koľko skutočne pozitívnych nálezov bolo naozaj nájdených. Pri výsledku klasifikácie je taktiež dôležitá, takzvaná F-Miera, taktiež známa ako balancované F-skóre, ktoré je harmonickým priemerom presnosti a senzitivity:

$$F = 2 \cdot \frac{\text{presnos} \cdot \text{senzitivita}}{\text{presnos} + \text{senzitivita}} \tag{4.1}$$

Všetky tieto informácii nám následne slúžia ako vyhodnocovací faktor presnosti detekcie.

[2] Smerodajná odchýlka je v teórií pravdepodobnosti a štatistike meradlom štatistickej disperzie.
[3] Entropia je fyzikálna veličina, ktorá meria mieru neusporiadanosti dát.
[4] Euklidovská vzdialenosť http://access.feld.cvut.cz/view.php?cisloclanku=2012090003

Kapitola 5

Implementácia príznakov

Ako bolo spomínané v kapitole 4, výber vhodných príznakov je kritický pre úspešnú klasifikáciu. V našom prípade detekcií aplikačných protokolov je vstupnými dátami sieťová premávka.

Vektor príznakov, ktorý budeme aplikovať na tieto dáta musí byť dostatočne obecný. Obecnosťou myslíme, to že príznak musí byť aplikovateľná na akúkoľvek dátovú vzorku. Nesmie sa stať že príznak sa bude zameriavať na špecifické charaktery sieťovej premávky, ktoré by favorizovali určitý výsledok. Tieto príznaky môžu následne pôsobiť rušivo, ktoré budú prispievať, ku takzvaným falošne pozitívnym výsledkom. Keďže sa náš vektor príznakov zameriava na dáta posielané po sieti, je nevyhnutné dopodrobna porozumieť premávke, ktorá sa na sieti vyskytuje.

Pri implementácií príznakov je potrebné zameriavať sa na štatistiky, ktoré nemôžu byť ovplyvnené výkonnosťou sieťových prvkov. Na referenčnom ISO / OSI modely[1] sú týmito vrstvami, fyzická, spojová a sieťová vrstva. Na týchto vrstvách sa vyskytujú informácie, ktoré hovoria o špecifických hodnotách, ktoré môžu byť ovplyvnené rôznymi typmi smerovačov, alebo zariadené cez, ktoré sieťová premávka putuje. Práve preto sa pri vytváraní funkcií sústreďujeme primárne na aplikačnú vrstvu ISO / OSI modelu. Aplikačná vrstva, ktorá je siedmou vrstvou umožňuje aplikáciám prístup ku komunikačnému systému a umožňuje tak ich spoluprácu. Do tejto vrstvy sa radia služby a protokoly, ako sú FTP, DNS, DHCP, POP3, SMTP, SSH.

5.1 Vybrané príznaky

Pri vytváraní funkcií príznakov, ktoré by odpovedali riešeniu problému detekcie aplikačných protokolov na ktorý sa zameriava táto bakalárska práca mi výrazne pomohla publikácia *Discriminators for use in flow-based classification*[14], ktorá obsahuje sadu príznakov, ktoré charakterizujú dátový tok na sieti. Každá z príznak sa zameriava na iný typ štatistík, ktoré môžeme získať z dát. Z tejto sady príznakov bola vybratá podmnožina, ktorú som následne implementoval. Teraz si predstavíme najzaujímavejšie z nich.

Čísla portov

Pri identifikácií aplikačných protokolov je číslo portu veľmi nápomocné. Funguje na princípe identifikácií na základe portov, o ktorej sme sa zmieňovali v sekcií 2.1. Číslo cieľového

[1] Referenční model ISO/OSI sa používa ako názorný príklad riešenia komunikácie v počítačových a telekomunikačných sieťach pomocou vrstevnatého modelu, kde sú jednotlivé vrstvy nezávislé a nahraditeľné.

portu, teda portu serveru nám môže predradiť o akú aplikačný protokol sa môže jednať. Ako bolo spomínané, čísla portov 0 až 1023 sú rezervované organizáciou IANA. Tieto čísla portov sú väčšinou využívané známymi službami, ako sú FTP(20/21), SMTP(25), HTTP(80), SSH(22) atď. Tieto skutočnosti môžu nasvedčovať tomu že číslo portu serveru je veľmi dobrou formou detekcie aplikačného protokolu. Bohužiaľ porty používané v sieťovej komunikácií môžu byť čísla portou dynamicky pridelené. Dynamické prideľovanie portov sa vyskytuje na strane klienta. Klient má prístup ku portov v rozmedzí 49452 až 65535, tieto porty sú mu náhodne pridelené.

Problémom pri detekcií portu servera je už spomínané smerovanie portov. Smerovanie, alebo mapovanie portov je funkcia NAT[2]. NAT je metóda remapovania jedného IP adresového priestoru na iný. Takto NAT dokáže zmeniť číslo portu v hlavičke paketu. Taktiež sa nemôžeme spoliehať na číslo portov pri P2P[3] komunikácií, ako je napríklad sťahovanie torrentov[4] používa dynamické priraďovanie portov na komunikáciu medzi členmi prenosu súborov. V našich metrikách používame čísla portu serveru a klienta. Ak sa čísla portov veľmi líšia, čo môže mať za následok práve mapovanie portov, tak je na váhach danej funkcie postarať sa o dôležitosť tejto štatistiky pri detekcií.

SYN, FIN a PUSH pakety

Príznaky, ktoré hovoria o existencií a počte SYN, FIN a PUSH paketov sú veľmi spoľahlivou detekciou TCP protokolu, nakoľko UDP ich neobsahuje.

SYN paket sa používa pri TCP trojcestný handshaku. Klient posiela žiadosť serveru o pripojenie použitím paketu Synchronize, server potvrdí túto žiadosť poslaním paketu SYN-ACK. V korektnom pripojené bez problémov by sa mal počet SYN paketov rovnať dvom. Aplikačné protokoly, ktoré používajú trojcestný handshake sú FTP, Telnet, HTTP, HTTPS, SMTP, POP3, IMAP, SSH a všetky ostatné protokoly, ktoré využívajú TCP. V niektorých prípadoch ako je metóda skenovania portov sa využíva, kde sa posiela iba SYN paket serveru a očakáva sa SYN-ACK odozva, ktorá hovorí o tom že zariadenie žije. Taktiež sa využívajú techniky DDOS pomocou záplavy SYN paketov.

FIN paket sa oproti SYN paketu používa na uzatvorenie spojenia. Zariadenie posielajúce FIN paket tým hovorí, že už nechce komunikovať s klientom.

PUSH paket je správou pre prijímajúci TCP zásobník, aby poslala tieto dáta ihneď prijímajúca aplikácií bez toho aby čakal kým sa zásobník naplní. Nastevenie PUSH paketu sa väčšinou nevykonáva posielajúcou aplikáciou, ale TCP vrstvou posielajúcej aplikácie. Moderné TCP/IP zásobníky nastavujú PUSH bit na konci bufferu pri používané funkcie *send()*. PUSH bit pomáha prijímajúcej strane optimalizovať výkonnosť pomocou zoskupovania dát do logických kusov. Ale nastávajú prípady kedy prijímajúca aplikácia potrebuje dáta z TCP vrstvy hneď ako prídu, namiesto čakania na v TCP vrstve na optimalizáciu vyrovnávacej pamäte. Takéto aplikácie sú napríklad streamovanie filmov a hudby, kde aplikácie prehrávajúca stream potrebuje dáta okamžite. Aplikácie používajúce nízke odozvy ako sú online hry, ktoré potrebujú byť zo synchronizované zo serverom. Väčšinou tieto aplikácie nevyužívajú TCP vrstvu, pretože príliž veľa optimalizácií na TCP vrstve môže mať za následok zhoršenie odozvy. Tieto aplikácie používajú UDP vrstvu a implementujú vlastne prenosové protokoly. Je možné PUSH bit odstrániť úplne, pre potreby optimalizácie. Z to-

[2]Network address translation

[3]Peer-to-peer

[4]Torrent je súbor, ktorý obsahuje metadáta o súboroch, ktoré budú distribuované. Obsahuje aj sieťové lokácie počítačov, ktoré napomáhajú nájsť umiestnenie súborov popísaných v torrent súbore.

hoto nám vyplýva že chýbajúca hodnota PUSH bitu môže značiť aplikácie z malou odozvou, UDP, alebo streamovacie aplikačné protokoly.

Typ transportného protokolu

Prístupom ku údajom na transportnej vrstve môžeme zistiť typ transportného protokolu. Najviac používané transportné protokoly sú TCP a UDP. Ostatné protokoly ako DCCP, SCTP a RSVP sú využívané menej. Táto informácia nám môže pomôcť pri detekcií aplikácií, ktorá využívajú špecifické transportné protokoly.

Segmentová časť

Segmentová časť, alebo v našej implementácií dĺžka payloadu na aplikačnej vrstve, je príznak, ktorá nám udáva veľkosť správy prenášanej v jednom TCP segmente. Do tejto štatistiky sa nepočíta TCP a ani IP hlavička. Väčší objem dát posielaných po sieti môže poukazovať na špecifické aplikačné protokoly, ktoré po sieti prenášajú veľa dát. Pri tomto príznaku počítame štatistiku z maximálnej a minimálnej hodnoty veľkosti segmentovej časti.

Kontrolne bajty

Kontrolnými bajtami v našom kontexte rozumieme veľkosť TCP hlavičky, pričom sa zameriavame na hodnoty od 20 hore. V týchto hodnotách, ktoré môžu mať veľkosť od 0 až 40 bajtov plus TCP hlavička a občasné zarovnanie (padding), sa skrývajú informácie o možnostiach paketu. Nachádza sa tu veľkosť maximálne povolenej veľkosti segmentu, selektívne povolenie potvrdenia, alebo TCP časová značka. Vo vektore funkcií počítajúcich metriky z kontrolných bajtov sa zameriavame na medián a vážený priemer, ktorých hodnoty nám môžu zlepšiť detekčné schopnosti. Následné hodnoty, ako sú maximum a minimum vyskytujúce sa v toku sú využiteľné pri detekcií rôznorodosti protokolov. Taktiež sa zameriavame na prvý a tretí kvartil, týchto hodnôt. Prvý kvartil nám hovorí o mediáne z prvej tretiny počtu kontrolných bajtov, teda môžeme usúdiť aká je bežná nižšia hodnota kontrolných bajtov. Tretí kvartil hovorí o mediáne vrchnej tretej tretiny počtu týchto bajtov. Z toho usúdime aká je bežná vyššia hodnota kontrolných bajtov.

Dĺžka spojenia

Táto funkcia, ktorá hovorí o dĺžke komunikácie klienta a serveru je jedna zo základných metrík používaných pri detekcií aplikačných protokolov. Dlhé spojenie môže identifikovať sťahovanie súborov, alebo pozerania videí. Naopak krátke spojenia môžu hovoriť jednoduchej komunikácií na sieti.

Zmena smeru toku

V priebehu života spojenia klienta a servera sa smer toku dát môže niekoľko krát zmeniť. Sledovanie počtu týchto zmien nám môže napovedať, ako často sa klient dotazuje na server, a ako často dostáva od servera odpoveď. Veľký počet zmien nasvedčuje protokoly, ktoré často komunikujú zo serverom ako napríklad HTTP.

Medzičas príchodu paketov

Medzičas príchodu dvoch po sebe nasledujúcich paketov. Počítame maximálny, minimálny čas a vážený priemer časov. Taktiež počítame prvý a tretí kvartil týchto hodnôt. Vážený priemer časov sa nazýva oneskorením. Ak je vážený priemer a medián týchto hodnôt s melou hodnotou štandardizovanej odchýlky hovoríme, že sa v toku nachádza malý jitter. Väčšie hodnoty hovoria a tom že jitter je prítomný. Oneskorenia na sieti môže napovedať o rôznych typoch aplikačných protokolov.

Dĺžka paketu

Dĺžka paketu nám udáva veľkosť celého paketu s hlavičkou a dátami. Veľkosti prázdnych UDP a TCP paketov sa líšia. Veľkosť celkového prázdneho TCP paketu je 64 bajtov, kdežto UDP je 52 bajtov.

Príznaky zameriavajúce sa na rozloženie bajtov v pakete

Funkcie, ktoré sa sústreďujú na postupnosť bajtov, ako sú Rovnosť prvých troch bajtov (Ako často sa prvé 3 bajty rovnajú. Toto je bežné v protokoloch s fixnou hlavičkou.), Heš prvých štyroch bajtov, alebo Pozícia prvého bitu v pakete. Tieto funkcie môžu strácať svoju výpovednú hodnotu pri šifrovanej komunikácií.

Ďalšie zaujímavé príznaky

- Frekvencia bajtov - frekvencia bajtov v payloade toku. Šifrované dáta sú rovnomerne rozložené,
- Počet bajtov,
- Počet paketov,
- Počet bajtov za jednotku času,
- Počet paketov za jednotku času,
- Distribúcia dĺžky paketov,
- Veľkosť prvého payloadu.

5.2 Testovanie príznakov

Pri implementácií funkcií bolo potrebné ich funkčnosť podrobne vyskúšať. Bolo potrebné zistiť či sú korektne naimplementované, teda či hodnoty, ktoré počítajú sú naozaj hodnotami, ktoré chceme zistiť. Pri implementácií funkcií bolo potrebné myslieť aj na validitu vstupov. Čo sa stane ak počítame veľkosť dát, ktoré paket neobsahuje?

V projekte bola naimplementovaná sada jednotkových testov, ktorá dopodrobna testovala každú funkciu. Pri testovaní bola veľmi dôležitá vzorka dát na, ktorých tieto jednotkové testy prebiehali. Bolo potrebné ručne vypočítať metriky pcapu[5], ktorý obsahoval zachytenú sieťovú komunikáciu.

[5]pcap - packet capture, je binárny súbor, ktorý má v sebe uložený záznam o sieťovej komunikácií a je možné z neho tieto údaje čítať.

Prvým krokom pri testovaní funkcií bolo získanie pcapu, ktorý bol vhodný na naše účeli. Keďže program Netfox Detective vyžaduje na fázu učenia sa pcap, ktorý bude obsahovať názvy aplikácií, ktoré vygenerovali sieťovú komunikáciu, tak sme museli pre zachytenie nášho testovacieho pcapu využi program Microsoft Network Monitor[6]. Tento program nám ku každej konverzácií pridá štítok s menom aplikácie, ktorá ho generovala. Toto nám dovoľuje využívať učiaceho algoritmu. Na naše účely bolo potrebné zachytiť tri konverzácia jedného druhu. Dve konverzácie slúžili na generovanie metrík a učenie sa. Tretia konverzácia slúžila na klasifikáciu. Základná sada testovacích dát bola vygenerovaná SSH konverzáciou. Pripojenie na vzdialený server pomocou programu Putty[7]. Pripojenie a odpojenie, ktoré generuje jednu konverzáciu sme opakovali tri krát.

Nad týmto pcapom sme následne ručne a za pomocí rôznych nástrojov ako je Wireshark, alebo tcpdump[8] vypočítali hodnoty výsledkov naimplemetovaných funkcií. Tieto výsledky sme použili ako hodnoty v našich jednotkových testoch. Pri implementácií jednotkových testov som čiastočne pracoval v skupine s Michalom Klčom, ktorý mal túto úlohu ako projekt na predmete PDS.

Následné odladovanie funkcií prebiehalo na integračných testoch na reálnych dátach, pri testovaní programu ako celku.

5.2.1 Váhy príznakov

Túto tému sme rozoberali v kapitole 4.2. Pri implementácií príznakov je potrebné ohodnotiť výsledky, ktoré vypočítame v priebehu učiacej fázy nášho programu. Dôležitosť každej štatistiky v aktuálnom kontexte je vyhodnotená pomocou váh. Táto váha hovorí o tom či je štatistika, pri detekcií dôležitá, a ako veľmi sa má algoritmus riadiť touto hodnotou. Váha je číslo s plávajúcou desatinou čiarkou, ktoré sa pohybuje v rozmedzí 0,0 - 1,0. Čím menšie je toto číslo, tým menšia váha je tejto príznaku priradená.

Spôsob počítania tejto váhy v tomto projekte je pomocou smerodajnej odchýlky, takzvanej STDEV. Smerodajná odchýlka je v štatistike reprezentovaná písmenom *sigma* (σ) a používa sa na výpočet štatistickej disperzie. Hovorí o tom ako široko sú rozložene hodnoty v množine. Smerodajná odchýlka je kladnou druhou odmocninou rozptylu. Vzorec na výpočet môžeme vidieť v rovnici číslo 5.1

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{5.1}$$

Strednú hodnotu smerodajnej odchýlky počítame, ako priemernú hodnotu zo všetkých hodnôt vypočítaných jednou funkciou príznaku pre danú konverzáciu. Z tejto strednej hodnoty následne počítame *stdev*. Výsledkom je hodnota, ktorá bude priradená ako váha.

Tieto váhy nemusia byť úplne presné a použiteľné, preto je potrebné niektoré váhy následne vybalansovať. Na to aby sme zistili ako fungujú váhy v našom programe a či po implementácií váh sa detekčné schopnosti naozaj zlepšili, potrebuje rozsiahle manuálne testovanie a upravovanie váh, tak aby boli váhy v kontexte našej detekcie prospešné.

Pri manuálnom testovaní sa vždy zameriavame na to, akú množinu dát poskytujeme programu. Podrobne kontrolujeme aké hodnoty naše metriky naberajú a aké váhy sú im

[6]Microsot Network Monitor https://www.microsoft.com/en-us/download/details.aspx?id=4865

[7]Putty je SSH a Telnet klienthttp://www.putty.org/

[8]tcp je program na zachytávanie sieťovej komunikácie na Linuxe, ktorý nám dáva možnosť filtrovania rôznych štatistických údajov.https://linux.die.net/man/8/tcpdump

pridelené. Keďže hodnota váhy sa pohybuje od 0,0 až po 1,0 je potrebné zaistiť aby sme eliminovali krajné hodnoty. Teda nesmie sa stať že bude veľa váh v hodnote 1,0, alebo 0,0. Tieto hodnoty môžu byť priradené váham iba pri špeciálnych prípadoch, ktoré by mohli pomôcť pri detekcií.

V tomto projekte bolo potrebné si dopodrobna naštudovať správanie sa protokolov pri rôznych formách komunikácie na sieti a podľa toho hľadať stavy, pri ktorých sa tieto protokoly správajú špecificky. Tieto informácie nám pomohli pri následnom prerozdeľovaní hodnôt. Ako príklad uvediem funkcie, ktoré počítajú veľkosť TCP hlavičky spolu bajtami pre nastavenie možností. Veľkosť TCP hlavičky je 20 bajtov a maximálny počet bajtov pre nastavenia TCP hlavičky je 40 bajtov. Ak veľkosť TCP hlavičky prekročí vyššie spomínaných 20 bajtov môžeme predpokladať že paket bude bližšie špecifikovať určitý typ sieťovej premávky, preto mu môžeme priradiť vyššiu dôležitosť pri detekcií, ako bežnému paket s veľkosťou hlavičky 20 bajtov. Z dokumentácie, ktorá popisuje tieto bajty sa môžeme dozvedieť že prvý bajt, hovorí o tom či sa nachádza zoznam možností alebo nie. Druhý bajt je bajtom zarovnávajúcim, pretože bajty možností musia byť deliteľné štyrmi. Nasledujúce štyri bajty hovoria o maximálnej veľkosti segmentu, ktorá sa môže nastavovať v určitých aplikačných protokoloch. Takto sme sa dostali ku ďalšej hodnote tejto metriky, ktorá môže mať väčšiu výpovednú hodnotu a to je veľko 26 bajtov. Túto hodnotu môžeme označiť, ako "dôležitejšiu", ako hodnotu 20 bajtov, preto jej nastavíme väčšiu váhu. Keď sme zistili medze pri, ktorých môže hodnota naberať väčšej dôležitosti musíme zistiť, pri akej metrike môže byť táto hodnota nápomocná. Funkcie, ktoré počítajú veľkosť TCP hlavičky počítajú jej priemernú veľkosť, maximálnu a minimálnu veľkosť, prvý a tretí kvartil. Priemerná veľkosť TCP hlavičky je okolo 20 bajtov, preto môžeme usúdiť že ak sa priemerná veľkosť zvýši nad hodnotu 22 bajtov môžeme priradiť väčšiu váhu tejto metrike. Pri maximálnej hodnote môžeme hovoriť o 26, alebo 32 a viac bajtoch. Prvý kvartil by sa mal väčšinou rovnať 20 bajtom, preto hodnoty, ktoré sú nad 20 bajtov môžeme nazvať dôležitejšími. Takýmto spôsobom sme postupovali pri implementovaní váh týchto funkcií. Niektoré funkcie potrebovali vybalansovať váhy pomocou manuálnych zásahov, niektoré fungujú adekvátne bez úprav.

Kapitola 4 rozoberala proces Feature Engineeringu. Popisovala tento proces ako zdĺhavý, vedecký, intuitívny a časovo náročný na testovanie. Táto časť práce bola časovo najnáročnejšia a stále má priestor pre zlepšenie. Algoritmus strojového učenia vďačí za svoje detekčné schopnosti hlavne funkciám, ktoré nazývame príznak.

5.3 Testovanie

Dôkladné testovanie tejto implementácie bolo veľmi zdĺhavé. V testovacej fáze sa ešte stále upravovali hodnoty váh funkcií, tak aby detekcia fungovala čo najlepšie. Každé nové testovacie dáta ponúkali dôležité informácie pre ďalšie úpravy váh. Pri testovaní boli využívané rôzne klasifikátory, ktoré sa učili na testovacích dátach a následne sa používali pri identifikácií.

Testovací proces bol následovný:

1. Zdrojom dát boli L7 konverzácie, ktorím bola pridelená informácia o procese, ktorý ich vygeneroval.

2. Dáta boli rozdelené na dve množiny dát podľa učiaceho sa pomeru. (0.7, 0.9), týmito pomermi sa snažíme zistiť výsledky detekcie pri veľmi vysokej miere učiacich sa dát.

3. Pre každú L7 konverzáciu v učiacej a testovacej množine bol vypočítaný vektor funkcií použitých na získanie štatistik danej konverzácie.

4. Vygenerované vektory funkcií boli zoskupené podľa informácií o aplikačnom procese a cieľovom porte transportnej vrstvy.

5. V trénovacej fáze sa generovanie klasifikátorov líši podľa použitej metódy:

   - Statistická metodá ESPI [9] - Model aplikačného protokolu bol vypočítaný pre každú skupinu.
   - Bayesova klasifikácia - Bayeskovský sieťový klasifikátor bol natrénovaný pre každú skupinu.

6. V testovacej fáze sú vytvorené klasifikátory použité na identifikáciu štítka triedy:

   - ESPI - Každý vektor funkcií symbolizujúci L7 testovaciu konverzáciu bol porovnaný s každým modelom aplikačného protokolu pomocou Euklidovskej vzdialenosti.
   - Bayesova klasifiácia - Každý Bayesovský klasifikátor poskytoval pravdepodobnosť, toho ktorá L7 testovacia konverzácia patrí do triedy reprezentovanej klasifikátorom.

7. Výsledkom klasifikácie bola sada vzdialeností, alebo pravdepodobností. Táto sada bola zoradená a odtlačok protokolu z najnižšou vzdialenosťou, alebo pravdepodobnosťou bol vybratý ako štítok.

8. Štítok bol následne porovnaný z anotáciou a boli vypočítané štatisticky klasifikácie.

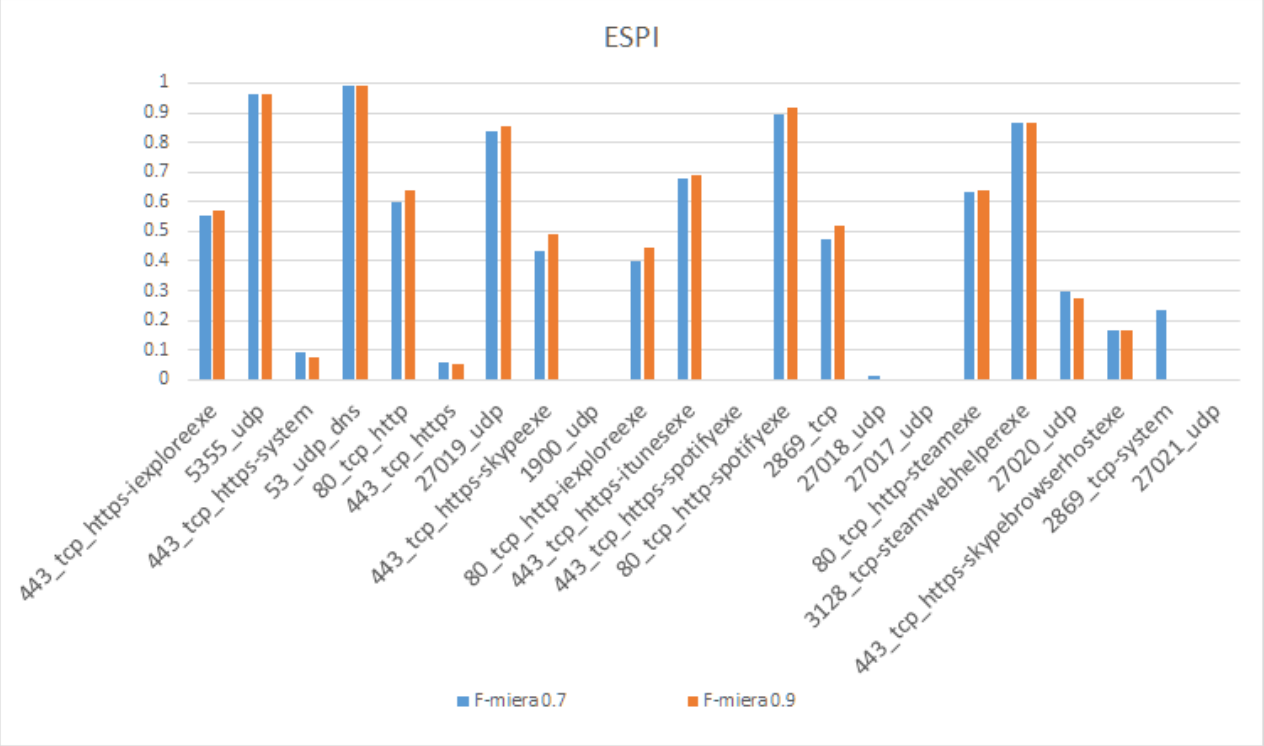
5.3.1 Testovanie pôvodnej funkcionality

Prvou časťou testovania bolo testovanie pôvodnej funkcionality implementácie. V pôvodnej implementácií bolo použité malé množstvo funkcií so základnou funkcionalitou. Zameriavali sme sa na porovnávanie výsledkov štatistickej metódy (ESPI) a metódy strojového učenia (ML[10]). Tieto dve metódy sú úplne rozličné ale sú založené na spoločnej sade príznakov, ktorá slúži pri identifikácií. Vykonávali sme dve testovania, kedy sme menili pomer učiacich sa dát. Testovali sme v pomeroch 0.7, 0.9. Porovnávali sme výsledky, ktoré nám poskytlo ESPI a ML. Metriky testovacích dát:

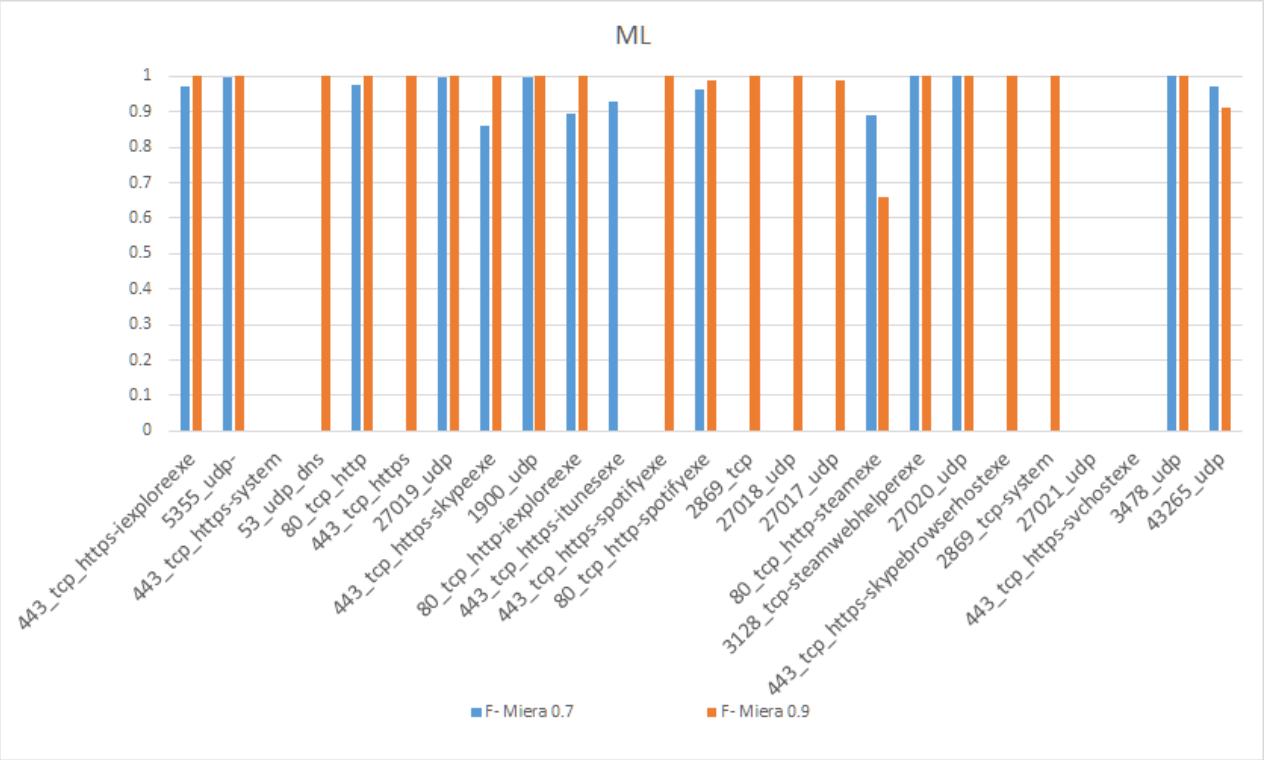
- Veľkosť súboru: 3290 MB,
- Formát: .cap,
- Časové rozpätia: 23:36:10,
- Počet paketov: 6192688,
- Počet L7 konverzácií: 74748,
- Počet L4 konverzácií: 82589,
- Počet L3 konverzácií: 240,

[9] ESPI - Echance Statistical Protocol Identification
[10] Machine Learning

- Počty označených konverzácií: Internet Explorer (13381), Skype (2899), iTunes(2306), Spotify (3285), Steam (1133).



Obr. 5.1: Počiatočná implementácia. ESPI s rôznym učiacim sa pomerom.



Obr. 5.2: Počiatočná implementácia. ML s rôznym učiacim sa pomerom.

Na obrázku 5.1 vidíme výsledky metódy ESPI. Pre potreby porovnávania výsledku používame F-Mieru.

Pohľad na výsledky klasifikácie na obrázku 5.1 ukazuje rozsiahly počet UDP tokov, ktorý má veľmi veľké rozptyly v presnosti detekcie. Môže to byť spôsobené tým že tento aplikačný protokol používa UDP veľmi podobným spôsobom a preto klasifikátor nedokázal dostatočne korektne určiť aplikáciu, ktorá tento protokol používa.

Na obrázku, ktorý predstavuje klasifikáciu pomocou strojového učenia 5.2 môžeme vidieť rovnaký jav ako na obrázku 5.1. Takto môžeme porovnať tieto dva obrázky, pretože obsahujú rovnaké dáta a kategórie.

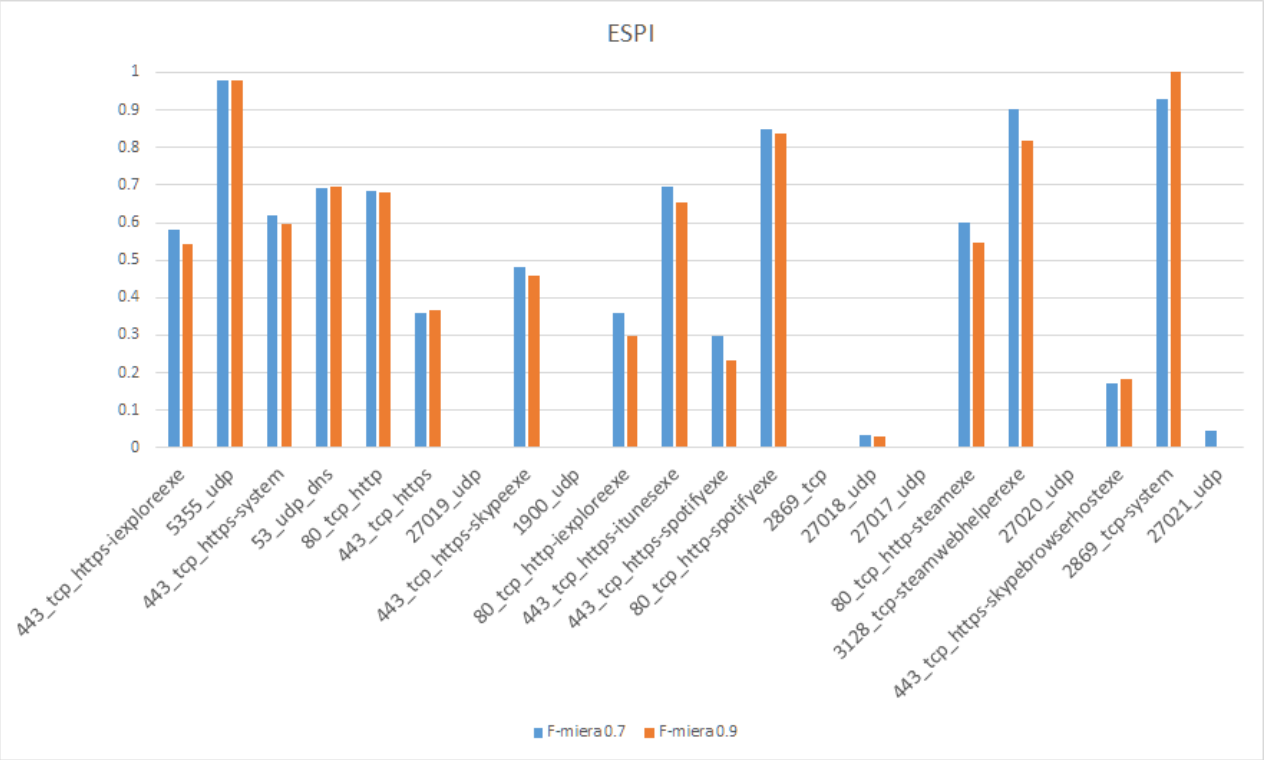
Bayesovký klasifikátor má problémy s klasifikáciou aplikácií, ktoré používajú protokoly podobným spôsobom. Na vyriešenie týchto nejasností pri detekcií môžeme použiť metódu hierarchického zhlukovania nad ESPI. Používateľ, takto môže manuálne zasahovať do tried aplikačných protokolov pomocou vizualizácie zhlukov. Agent Netfox Detective obsahuje metriky úspešnosti, ktoré sú popísané v kapitole 4.3. Tieto metriky používame pri detekcií ako štatistické hodnoty úspešnosti klasifikácie. Príklad týchto metrík môžeme vidieť v tabuľke 5.1, ktorá ukazuje výsledky klasifikácie pomocou štatistickej metódy s pomerom učiacich sa dát 0,7 a klasifikáciou podľa čísla portu a názvu protokolu. Keďže táto klasifikácie neobsahuje mená aplikácií zameriava sa na obecné identifikovanie protokolov použitých v sieťovej premávke.

Tabuľka 5.1: Tabuľka EPI klasifikácie podľa čísla portu a protokolu.

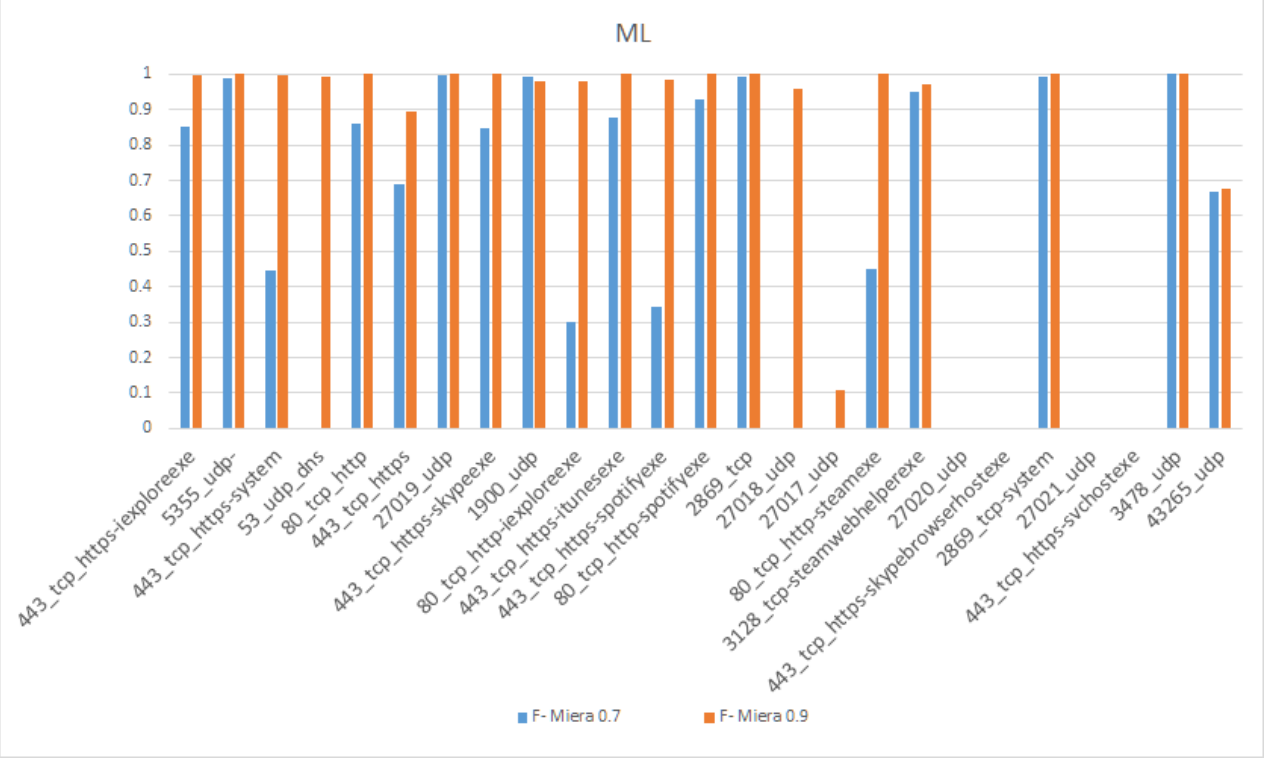
Štítok aplikácia	TP	FP	FN	Presnosť	Nález	F - Miera
138_udp	27	0	0	1.000	1.000	1.000
40029_udp	11	1	5	0.917	0.688	0.786
139_tcp	20	0	0	1.000	1.000	1.000
27019_udp	731	108	177	0.871	0.805	0.837
27018_udp	9	61	163	0.129	0.052	0.074
27017_udp	2	11	166	0.154	0.012	0.022
67_udp_dhcps	24	0	0	1.000	1.000	1.000
27020_udp	115	514	16	0.183	0.878	0.303
43265_udp	36	3	6	0.923	0.857	0.889
27021_udp	1	0	61	1.000	0.016	0.032
137_udp	1	0	25	1.000	0.038	0.074
80_tcp_http	3209	179	32	0.947	0.990	0.968
3478_udp	47	0	0	1.000	1.000	1.000
1900_udp	0	0	859	0.000	0.000	0.000
3128_tcp	96	3	51	0.970	0.653	0.780
2869_tcp	219	0	76	1.000	0.742	0.852
5351_udp	30	494	0	0.057	1.000	0.108
443_tcp_https	9248	29	53	0.997	0.994	0.996
53_udp_dns	2358	54	9	0.978	0.996	0.987
443_udp_https	19	2	4	0.905	0.826	0.864
3702_udp	17	0	0	1.000	1.000	1.000
5355_udp	3295	243	0	0.931	1.000	0.964
4070_tcp	16	2	1	0.889	0.941	0.914

5.3.2 Testovanie implementácie

Testovacie dáta použité pri testovaní tejto bakalárskej práce obsahovali 25GB dátovej komunikácie zachytenej vo formáte pcpap. Sieťová komunikácia bola zachytená na školských počítačoch študentom Dušanom Drevickým, ktorá mi bola poskytnutá Ing. Jánom Pluskalom, na potreby testovania. V testovacej sade sú dáta z ôsmych zariadení, ktoré mali nainštalovaný operačný systém Windows 7 a boli pripojené ku internetu cez proxy server. Každý pcap s týchto testovacích dát obsahoval aspoň tridsať rovnakých konverzácií pri používaní každej aplikácie. Zber dát prebiehal na často používaných aplikáciach, ako sú Spotify, Skype, Steam, iTunes a iné. Pri generovaní jednotlivých konverzácií sme postupovali následovne: Zapnúť Skype, prihlásiť sa, napísať spravu/poslať súbor/zavolať, vypnúť Skype. Tento postup nám vygeneroval kompletnú konverzáciu. Tento postup sa opakoval 30 krát pre všetky aplikácie, ktorých komunikáciu sme zaznamenávali. Pri zaznamenávaní komunikácie sme používali výhradne Microsot Network Monitor, ktorý nám označkoval komunikáciu číslom a menom aplikácie, ktorá tento tok dát využívala. Dáta boli zaznamenané z inštalácie aplikácií a bežného používania. Veľkosti pcapov sa sa pohybovali od 30MB až po 600MB. Tieto pcapi obsahovali veľké množstvo šifrovanej komunikácie, ktoré bola taktiež cieľom testovania. Z tohoto množstva dát sme si pre potreby nášho testovania vybrali 3GB dát na, ktorých sme uskutočnili záverečné testovanie.



Obr. 5.3: Nová implementácia. ESPI s rôznym učiacim sa pomerom.



Obr. 5.4: Nová implementácia. ML s rôznym učiacim sa pomerom.

Obrázky 5.3 a 5.4 zobrazujú výsledky základných testov, ktoré sme uskutočnili po rozšírení sady štatistických funkcií na rovnakej testovacej vzorke ako v podsekcií 5.3.1. Z výsledkov na obrázku 5.3, kde sme sa zamerali na ESPI, pozorujeme že spoľahlivosť klasifikácie aplikácií, ako je napríklad *Internet Explorer*, ktorého počet konverzácií je v našich dátach najpočetnejší (10998). Tento údaj si môžeme nájsť v prílohe B v tabuľke B.1. Klasifikačné schopností, ktoré popisuje F - Metrika sa pohybujú nad hodnotou 0.5 pri spomínanej aplikácií *Internet Explorer*. Treba brať do úvahy že táto komunikácie je šifrovaná, čo nám môže spôsobiť menšie problémy pri detekcií. Tento výsledok je porovnateľný z výsledkom originálnej implementácie, kde sa hodnota pohybovala približne v rovnakých rovinách čo môžeme vidieť na grafe 5.1. *5355 UDP* je ďalším protokolom z počtom konverzácií presahujúcim 10000. Klasifikácia tohoto protokolu sa pohybuje veľmi vysoko v hodnotách blížiacim sa 100% pravdepodobnosti. Výrazný skok v detekcií je viditeľný na aplikácií *system* používajúcej port 443 a šifrovanú TCP komunikáciu. Hodnota sa z predchádzajúcej implementácie, ktorá dosahovala presnosti pod 10% dostala na hranicu 60%. Taktiež vidíme aj rozdieli, ktoré hovoria v neprospech veľkému počtu naimplementovaných príznakov. Môžeme si všimnúť že hodnota presnosti v tejto sade testovacích dát aplikácie *53-udp-dns* klesla z hodnoty blížiacej sa 100% na hodnotu približne 70%. V udájov vyplíva, že hodnoty detekcie kolísajú v rozmedziach predchádzajúcich hodnôt na veľkom počte dát. Z menším počtom dát kolíše aj presnosť týchto klasifikačných funkcií. Preto pri testovaní sme sa sústredovali na aplikácie, ktorých počet oštítkovaných komunikácí bol väčší ako 50. Hlavným problémom sa naďalej stávajú aplikácie, ktoré používajú rovnaký protokol podobným spôsobom. Tento jav je očividný na príklade protokolov *5355-udp* a *1900-udp*, aplikácia využívajú protokol UDP na porte 5355 obsahuje 10980 konverzácií a jej presnosť klasifikácie sa blíži 100%, kdežto *1900-udp* z počtom tokov 2862 je v obidvoch prípadoch nulová, ako na grafe popisujúcom novú 5.1, tak aj starú implementáciu 5.3. V tabuľke 5.1 ukazujúcej metriky úspešnosti uvidíme že počet Falošne negatívnych klasifikácií bol 859 a nevyskytovali sa tu žiadne skutočne pozitívne ani falošne pozitívne prípady klasifikácie.

Tabuľka 5.2 priamo porovnáva výsledky štatistickej metódy a metódy strojového učenia, pri rôznych pomeroch trénovaných dát. Z tabuľky je jasné presnosť obidvoch metódy, ktoré čerpajú dáta z príznakov, ktoré boli rozšírené v rámci tejto práce je priamo úmerná počtu dát, ktoré musí aplikácia spracovať pred samotnou klasifikáciou. Presnosť klasifikácie teda závisí od počtu dát. Aj keď počiatočná implementácia obsahovala síce malú množinu funkcií na výpočet charakteristík tokov, jej presnosť pri dostatočnom počte dát dosahovala uspokojivých výsledkov. Nová implementácia rozšírila tieto charakteristiky, ktoré sú jadrom metódy strojového učenia a štatistickej metódy. Rozšírenie by malo priniesť obecnejšie detekčné schopnosti týchto metód a otvára dvere novým metódam na zlepšenie klasifikácie, akou môže byť automatický výber príznakov, ktoré nemusia byť pri danej klasifikácií relevantné,ale metóda predprípravených sád príznakom pre rozdielne klasifikačné prípady.

Tabuľka 5.2: Porovnanie presnosti ESPI a ML pri učiacich sa pomeroch 0,7 a 0,9

Štítok aplikácie	ESPI		ML		
	F-miera 0.7	F-miera 0.9	F-miera 0.7	F-miera 0.9	L7 Konverzácie
137_udp	0.074	0.000	0.080	0.000	84
138_udp	0.426	0.720	1.000	1.000	87
139_tcp-system	0.947	0.923	0.000	0.000	66
1900_udp	0.000	0.000	0.992	0.980	2862
27017_udp	0.000	0.000	0.000	0.105	557
27018_udp	0.033	0.032	0.000	0.961	571
27019_udp	0.000	0.000	0.998	1.000	3026
27020_udp	0.000	0.000	0.000	0.000	435
27021_udp	0.046	0.000	0.000	0.000	204
2869_tcp	0.000	0.000	0.992	1.000	656
2869_tcp-system	0.929	1.000	0.995	1.000	325
3128_tcp-steamwebhelperexe	0.900	0.819	0.951	0.965	489
3478_udp	0.638	0.609	1.000	1.000	156
3702_udp	0.791	0.706	0.000	0.000	56
40029_udp	0.286	0.308	0.500	0.910	53
4070_tcp-spotifyexe	0.882	1.000	0.000	0.000	55
43265_udp	0.689	0.643	0.667	0.678	139
443_tcp_https	0.360	0.365	0.689	0.894	3307
443_tcp_https-iexploreexe	0.582	0.543	0.851	0.998	10998
443_tcp_https-itunesexe	0.696	0.653	0.879	1.000	2178
443_tcp_https-skypebrowserhostexe	0.172	0.182	0.000	0.000	385
443_tcp_https-skypeexe	0.482	0.460	0.849	1.000	2899
443_tcp_https-spotifyexe	0.299	0.233	0.343	1.000	1640
443_tcp_https-steamexe	0.000	0.000	0.000	0.986	61
443_tcp_https-steamwebhelperexe	0.000	0.000	0.000	0.000	54
443_tcp_https-svchostexe	0.239	0.091	0.000	0.000	171
443_tcp_https-system	0.621	0.596	0.445	0.998	9309
443_udp_https	0.000	0.000	0.000	0.000	74
53_udp_dns	0.694	0.696	0.000	0.994	7888
5351_udp	0.811	0.769	0.991	0.000	98
5355_udp	0.977	0.978	0.990	1.000	10980
67_udp_dhcps	1.000	1.000	0.950	1.000	79
80_tcp_http	0.685	0.679	0.861	1.000	6173
80_tcp_http-iexploreexe	0.359	0.297	0.302	0.979	2383
80_tcp_http-itunesexe	0.000	0.000	0.000	0.000	128
80_tcp_http-spotifyexe	0.848	0.838	0.931	1.000	1590
80_tcp_http-steamexe	0.599	0.548	0.451	1.000	529

Kapitola 6

Záver

Cieľom tejto práce bolo rozšírené modulu rozpoznávania aplikačných protokol agenta Netfox detective so snahou o spresnenie jeho detekčných schopnosti.

V rámci teoretickej časti tejto práce bolo potrebné zoznámiť sa z problematikou identifikácie aplikačných protokolov za pomoci metód strojové učenia a štatistických metód, ktoré sa v tomto programe využívajú. Bayesovká klasifikácia, ktorá zastupuje metódu strojové učenia s učiteľom a klasifikácia na základe štatistik sieťovej komunikácie sú hlavnými detekčnými metódami. Následne som si naštudoval fungovanie týchto algoritmov vo frameworku. Netfox Detective je dynamický rozvíjajúci sa veľký projekt, ktorého podrobnú funkčnosť som si musel odskúšať pomocou krokovania behu programu, aby som dopodrobna pochopil jeho fungovanie.

Za metódu vhodnú na vylepšenie klasifikácie agenta sme zvolili metódu Feature Engineering. Táto metóda výživa poznatky o subjekte, ktorý identifikujeme na to aby vytvorila sadu funkcií príznakov, ktorá sa používa na vytvorenie charakteristík subjektu, v našom prípade sieťovej premávky. Po naštudovaní charakteristík, ktoré sú nápomocné pri detekcii aplikačných protokolov sa zvolila množina príznakov, ktoré boli následne implementované. Implementácia bola dôkladne otestovaná pomocou jednotkových testov. Nasledujúcim krokom bolo upravenie výpočtov váh jednotlivých príznakov, a iteratíve testovanie so snahou o zlepšenie identifikačných schopností. V poslednej fáze práce boli vybrané vhodné súbory zo zachytenou sieťovou premávkou, na ktorých prebiehalo rozsiahle testovanie funkčnosti detekcie. Výsledky sme porovnali s predchádzajúcou implementáciou v poslednej časti tejto práce.

Problémové časti práce sa vyskytovali hlavne vo fáze finálneho testovania. Testovanie na veľkom objeme dát je veľmi náročne na výpočetnú techniku a taktiež veľmi zdĺhavé. Neustály aktívny vývoj frameworku mohol občas prispieť ku zdržaniu pri implementácií, alebo testovaní.

Výsledne identifikačné schopnosti programu sa výrazne nelíšili. Kolísali v medziach predchádzajúcej implementácie. V niektorých prípadoch sa klasifikačné schopnosti spresnili, alebo zobecnili. V niektorých sa pohybovali na rovnakých, alebo nižších hodnotách ako v predchádzajúcej implementácií. Keďže príznaky sú jadrom detekčných schopnosti algoritmu strojového učenia a štatistickej metódy rýchlo vyvíjajúceho sa programu Netfox Detective je možné na túto rozšírenú implementáciu nadviazať.

Literatúra

[1] Bishop, C.: *Pattern recognition and machine learning*. 2006, ISBN 0-387-31073-8.

[2] Burges, C. J. C.: A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, ročník 2, 1998: s. 121–167.

[3] Cascarano, N.: Application Layer Traffic Classification. 2007.
URL "http://netgroup.polito.it/Members/niccolo_cascarano/pub/tesi.pdf"

[4] Ghoshal, A.; Povey, D.: Sequencediscriminative training of deep neural networks. In *in Proc. INTERSPEECH*, 2013.

[5] Griffin, K.; Schneider, S.; Hu, X.; aj.: Automatic Generation of String Signatures for Malware Detection.

[6] Haffner, P.; Sen, S.; Spatscheck, O.; aj.: ACAS: Automated construction of application signatures. In *In SIGCOMM'05 MineNet Workshop*, 2005.

[7] Hjelmvik, E.: The SPID Algorithm: Statistical Protocol IDentification. 2008.

[8] I, B.-G.: Bayesian Networks.
URL "http://www.eng.tau.ac.il/~bengal/BN.pdf"

[9] Jain, A. K.: Data Clustering: 50 Years Beyond K-Means. 2008.

[10] Jhakhar, A.: An Improvement to the K-Means and K-MEDOID using D-M Clustering Approach.

[11] Jiawei Han, M. K.: Data Mining: Concepts and Techniques. 2000.

[12] Lawrence, S.; Giles, C. L.; Tsoi, A. C.; aj.: Face Recognition: A Convolutional Neural Network Approach. *IEEE Transactions on Neural Networks*, ročník 8, 1997: s. 98–113.

[13] Li, W.; Moore, A. W.: A Machine Learning Approach for Efficient Traffic Classification. In *in Proceedings of the IEEE MASCOTS*, 2007.

[14] Moore, A.; Crogan, M.; Moore, A. W.; aj.: Discriminators for use in flow-based classification. Technická zpráva, 2005.

[15] Moore, A. W.; Papagiannaki, K.: Toward the accurate identification of network applications. In *In PAM*, 2005, s. 41–54.

[16] Moore, A. W.; Zuev, D.: Internet traffic classification using bayesian analysis techniques. In *In ACM SIGMETRICS*, 2005, s. 50–60.

[17] Ou, G.: Understanding Deep Packet Inspection (DPI) Technology3.
URL "http://www.digitalsociety.org/files/gou/DPI-Final-10-23-09.pdf"

[18] Pachghare, M. V. K.; Kulkarni, D. P.: Network Security Based On Pattern Matching: An Overview.

[19] Quinlan, J. R.: Induction of Decision Trees. *MACH. LEARN*, ročník 1, 1986: s. 81–106.

[20] Ruggieri, S.: Efficient C4.5. 2000.

[21] Schapire, R.: Machine Learning Algorithms for Classification.
URL "http://www.cs.princeton.edu/~schapire/talks/picasso-minicourse.pdf"

[22] Schneider, P.: *TCP/IP Traffic Classification Based on Port Numbers*. Dizertační práce, Harvard University, 1996.
URL "http://www.schneider-grin.ch/media/pdf/diploma_thesis.pdf"

[23] Sebastian Zander, T. N.; Armitage., G.: Automated traffic classification and application identification using machine learning. 2005.

[24] Sharma, T.; Sinha, K.: Intrusion Detection Systems Technology.

[25] Shu, G.; Lee, D.: Network Protocol System Fingerprinting – A Formal Approach. In *Proceedings of IEEE Infocom*, 2006.

[26] Sig, F.; Deng, L.; Y, D.; aj.: Deep Learning Methods and Applications.

[27] Smola, A.; Vishwanathan, S.: *Introduction to Machine Learning*. Press Syndicate of the University of Cambridge, 2005, ISBN 0-521-82583-0.

[28] Stephenson, T. A.: An Introduction to Bayesian Network Theory and Usage. 2000.

[29] Yao, X.: Evolving Artificial Neural Networks. 1999.

[30] Zander, S.; Nguyen, T.; Armitage, G.: Self-learning IP Traffic Classification based on Statistical Flow Characteristics. In *In proceedingof the sixth passive and active measurement workshop*, 2005.

[31] Zhao, H.: The Application and Research of C4.5 Algorithm.

Prílohy

Príloha A

Iné metódy strojového učenia

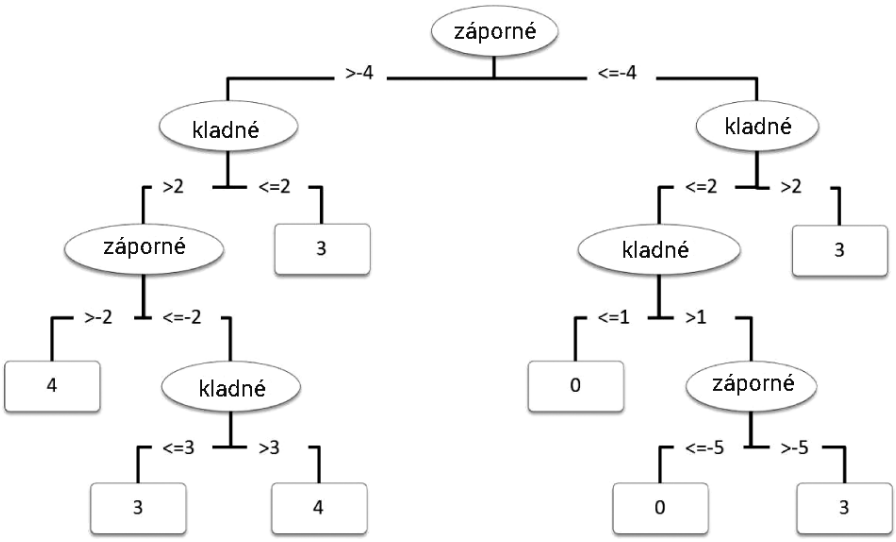
A.1 Učenie s učiteľom - Supervised learning

A.1.1 Rozhodovacie stromy

Model rozhodovacích stromov má podobu vývojového diagramu odvodeného od stromovej štruktúry, v ktorej každý vnútorný (nelistový) uzol predstavuje test určitej vlastnosti, pričom hrana vyjadruje výsledok tohoto testu[19]. Každý listový uzol obsahuje názov ďalšej triedy. Klasifikácia pomocou rozhodovacích stromov prebieha od koreňového uzlu a končí na uzle listovom. Binárny strom je taktiež rozhodovacím stromom. Každá cesta stromom vytvára množinu charakteristík a vlastností danej triedy, ktorú môžeme ďalej interpretovať ako odtlačok. Tento algoritmus vyžaduje predom klasifikované dáta pre vytvorenie modelu. Z tejto množiny je následne odvodená stromová štruktúra. Pri vytváraní stromu je podstatné rozlišovať, aké hodnoty môže dáta nadobúdať[11][19]:

- Spojité hodnoty: Z uzlu testujúceho daný atribút pôjdu iba dve hrany. Je potrebné si určiť deliaci bod, podľa ktorého určíme, ktorú cestu v stromovej štruktúre je potrebné si zvoliť. Napríklad ak budeme mať vlastnosť X a chceme zistiť či je jej hodnota kladná alebo záporná, tak zvolíme deliaci bod 0 a hrana na jednej strane bude označovať kladné hodnoty a hrana na druhej strane záporné hodnoty.
- Diskrétne hodnoty: Pri tvorbe stromu povedie z nelistového uzlu predstavujúceho test dané vlastnosti, toľko hrán, koľko je počet možných hodnôt. Ak budeme mať atribút X, ktorý môže nadobúdať hodnôt jedna, dva, tri, tak z daného uzlu povedú práve tri hrany.
- Diskrétne hodnoty v binárnom strome: Z každého nelistového uzlu môže viesť maximálne dve hrany, ale diskrétnych hodnôt môže byť niekoľko násobne viac. Riešením v tomto prípade je test či daný atribút spadá do určitej podmnožiny z hodnou, ktoré môže atribút nadobúdať. Ak bude náš atribút nadobúdať hodnoty jedna až štyri, tak sa test bude pýtať, či zdokumentovaný atribút patrí do podmnožiny zloženej z hodnôt jedna a tri.

Samotná klasifikácia pozostáva z priechodu stromom a testovania jednotlivých vlastností dát, ktoré chceme klasifikovať. Klasifikácia spolu s učiacim procesom dá sú rýchle, optimalizované a dosahuje dostatočne veľkej presnosti. Jednou z ich výhod je udržovanie viac dimenzionálnych dát. Výsledky klasifikácie veľmi úzko súvisia s učiacou množinou.



Obr. A.1: Jednoduchrozhodovací stroj

Na obrázku A.1 vidíme jednoduchý rozhodovací strom so spojitými hodnotami.

C4.5

Algoritmus bol vytvorený pre potreby vytvárania modelov rozhodovacích stromov pomocou metódy rozdeľ a panuj. Implementácia je veľmi rozšírená a obľúbená. Dosahuje veľmi veľkej presnosti a malej chybovosti. Mal zdokonaliť slabé stránky algoritmu ID3. Jedným s limitujúcich faktorov ID3 je jeho prílišná citlivosť na atribúty obsahujúce veľký počet hodnôt. Na prekonanie tohoto problému používa C4.5 takzvaný informačný nárast. Táto hodnota hovorí a pomere rastu. Pomer rastu je definovaný v rovnici A.1[31]:

$$PomerRastu(p,T) = \frac{Zisk(p,T)}{DeliacaInformacia(p,T)} \tag{A.1}$$

Člen DeliacaInformacia je popísaný v rovnici A.2:

$$DeliacaInformacia(p,test) = -\sum_{j=i}^{n} p\prime(\frac{j}{p}) \times \log\left(P\prime\frac{j}{p}\right) \tag{A.2}$$

Pri vytváraní stromu potom s premennými pracujeme rovnako, ako v prípade rozhodovacích stromov. Je potrebné rozlišovať či pracujeme so spojitými alebo diskrétnymi dátami. Avšak pri implementácií môžeme ku numerickým dátam nadobúdajúcich niekoľko málo hodnôt pristupovať ako ku dátam spojitým a pri teste ich rozdeliť deliacim bodom. Tento bod môžeme najčastejšie získať zaradením množných hodnôt atribútu a určením prahu pre každú susediacu dvojicu.

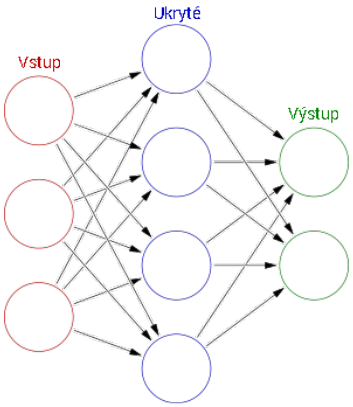
Takto zostavený strom môžeme testovať a prerezávať (Tree prunning). Vďaka prerezávaniu stromu môžeme nahradiť veľmi chybový podstrom. Najčastejšie ho nahrádzame listovým uzlom s najviac sa vyskytujúcou triedou, alebo ho nahradíme iným podstromom[20].

A.1.2 Neuronové siete

Neurónová sieť je výpočetný model, ktorý je zostavený na základe abstraktu vlastností biologických nervových systémov. Hlavnou zložkou a základnou časťou neurónovej siete je model neurón s N vstupmi a M výstupmi, ktorý spracováva informácia podľa pravidla A.3.

$$o_i^{k+1} = f(\sum_{j=1}^{N} w_{ij}^k \times o_j^k - \Theta i^{k+1}) \tag{A.3}$$

kde $0 < i \leq M$, $0 < j \leq N$, o_i^{k+1} je výstupná hodnota i-teho neuronu k+1 vsrtvy, k je index vsrtvy, Θi^{k+1} je prah excitácie i toho neurónu k+1 vrstvy ,w_{ij}^k je váha spojenia medzi j-tým neurónom k vrstvy a i-tým neurónom k+1 vrstvy a $f()$ je ľubovoľná monotónna funkcia.



Obr. A.2: ANN model

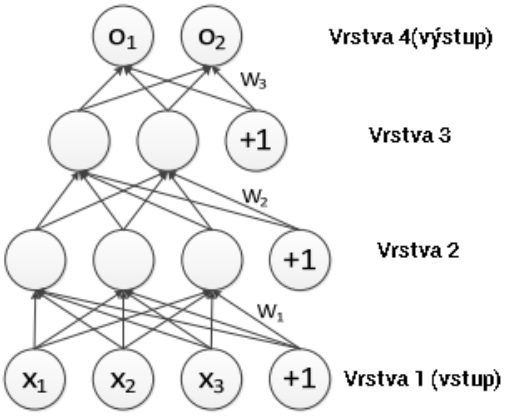
Neurónová sieť je poskladaná s viacerých vrstiev s rozličným počtom neuronou, prepájaních rôznymi spôsobmi, ako je možné vidieť na obrázku A.2 [29].

Najpodstatnejšou vlastnosťou neurónových sieti je schopnosť abstrakcie pravidiel medzi vstupnými a výstupnými hodnotami prezentovanými vo vhodnej forme. Pravidlá následne aplikujeme na akékoľvek vstupné hodnoty. Neurónové site sa využívajú v regulačnej a simulačnej technike.

Proces abstrakcie pravidiel sa nazýva učenie. Počas tohto procesu sa aktualizujú hodnoty váhových spojení a po ukončení učenia sa uť hodnoty nemania a sieť produkuje výstupy na základne aplikovaného pravidla na vstupné hodnoty. Výhodou takéhoto prístupu je paralelne spracovanie informácii, nepotrebnosť informácie o štruktúre procesu na, ktorý sa aplikovaný, taktiež možnosť adaptácie na zmenu parametrov. Je veľmi vhodná na identifikácie, aproximácie, klasifikácie a triedenie vzorov a sú univerzálnym aproximátorom, schopným aproximovať akúkoľvek spojitú funkciu s ľubovoľnou presnosťou.

Artificial Neural Network

Algoritmus ANN je algoritmom strojového učenia, ktorý inšpirovaný biologickými neurónovými sieťami v mozgu. Tento algoritmus je rozšírený v oblasti Pattern Recognition. Jedným z klasických využití je klasifikácia obrázkov, rozpoznávanie reči, kategorizácia textu a mnoho iných. Všeobecný ANN model je zobrazený na obrázku A.3.



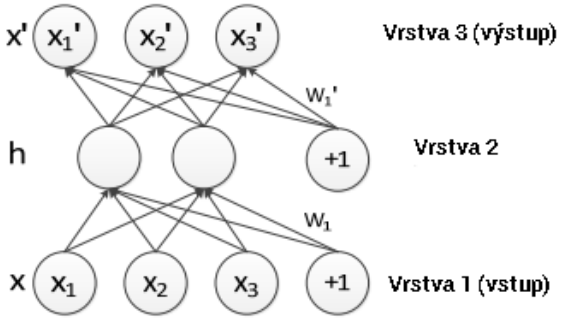
Obr. A.3: ANN model

Ako je znázornené na obrázku, ANN je reprezentovaný ako systém predvýpočtu neurónov vo viacerých vrstvách. Každý pár susediacich vrstiev je prepojený, zatiaľčo neuróny na rovnakej vrstve nemajú žiadnu spojitosť. Neurónmi nazývame uzly tejto štruktúry. Výsledkom ANN je optimalizácia váhového parametra medzi dvomi susediacim vrstvami a predvídanie tohoto parametra. Model s optimálnym parametrom následne použijeme ako reálne dáta[2].

Deep Learning

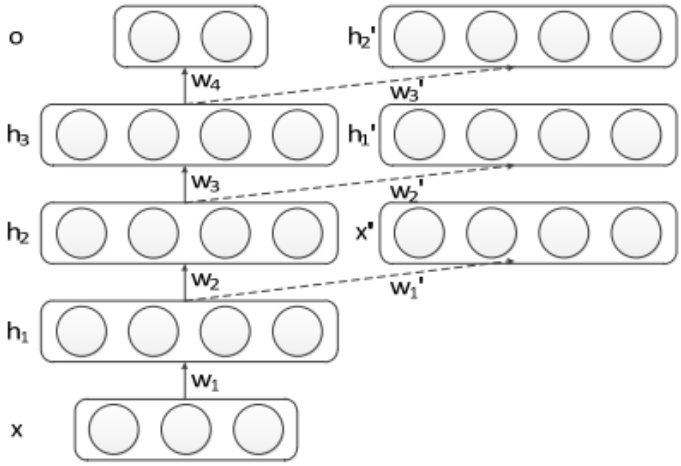
Deep learning[26], alebo Metóda hlbokého učenia je odvetvím strojového učenia založeného na vektore algoritmov. Niektoré známe algoritmy zahrnuté v tomto vektore sú Deep Neural Networks (DNN)[4], Convolutional Neural Networks (CNN) [12], Deep Belief Networks(DBN) a Stacked Auto-Encoder (SAE). V posledných rokoch popularita algoritmu hlbokého učenia stúpla a dostal sa do odvetví počítačového videnia, rozoznávania reči a prirodzeného pracovania reči. Neustála práca na inováciach tento sady algoritmov viedla ku poklesu chybovosti z 26% na 15% v ImageNet Challenge 2012.

Jednou z najdôležitejších vlastností tejto metódy je implementácia efektívnych algoritmov pre učenie bez učiteľa a hierarchickú extrakciu funkcií. Autoencode je perfektným príkladom. Tento prvok sa učí efektívne reprezentovať sadu komprimovaných dát. Štruktúru autoencodera môžeme vedieť na obrázku A.4:



Obr. A.4: Schéma autoencodera

V podstate je autoencoder iba formou ANN. Počet vrstiev sa vždy rovná trom. Rozdielom je iba vtom že uzly vo výstupnej vrstve sú rovnaké ako vo vstupnej vrstve. Uzly v strednej vrstve sú nové charakteristiky a sú nízko dimenzionálnou reprezentáciou. To znamená že dáta môžu byť zrekonštruované po tom ako budú použité v zložitých výpočtoch. Proces trénovania nezahŕňa žiadne predom klasifikované dáta, takže patrí ku učeniu bez učiteľa. Zbieranie dát bez klasifikácie je veľmi praktické. Ak odpadá potreba cvičných dát, nieje je potrebný ani čas a prostriedky na ich zber a klasifikáciu. Toto je jeden z dôvodov prečo je autoencoder tak populárny.



Obr. A.5: Schéma SAE

Zhromaždením viacerých štruktúr môžeme vytvoriť hlboku sieť, ktorá je vyobrazená na obrázku A.5. Každý výsledok trénovania strednej vrstvy je zoradený kaskádovo na obrázku. Sieťová štruktúra s takýmto rozložením sa volá Stacked Auto-Encoder (SAE). Pri dostatočnom objeme klasifikovaných dát, môžeme tieto dáta pripojiť do štruktúry na novú vrstvu a to mán dovolí trénovať model precíznejšie a zvýšiť jeho presnosť.

A.1.3 Algoritmus KNN

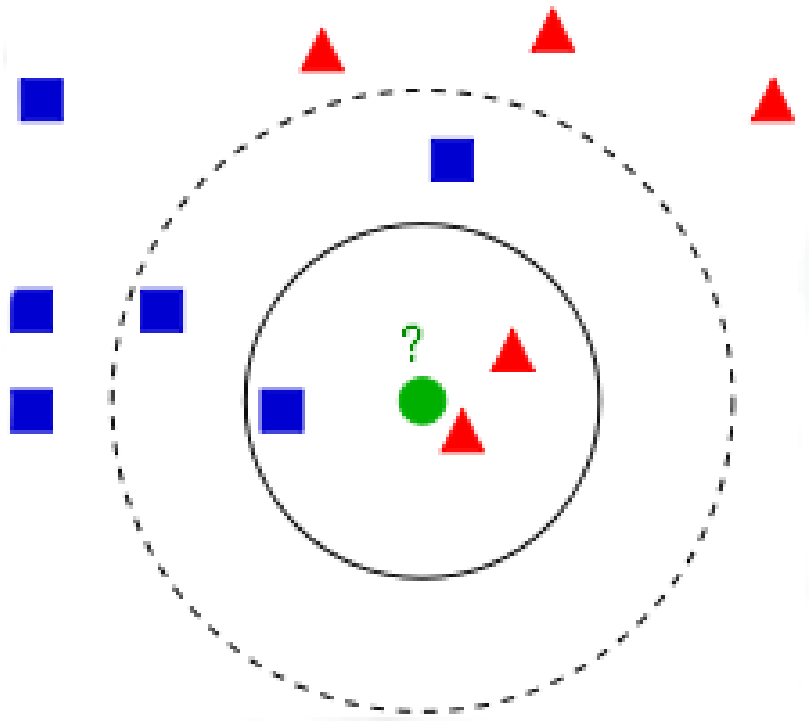
Algoritmus KNN, alebo aj algoritmus k-najbližších susedov patrí do skupiny takzvaného lenivého učenia. Špecifickou vlastnosťou tejto skupiny algoritmov je že sa model pre klasifikáciu vytvára čo najneskôr a to je vtedy až vtedy keď má ku dispozícií dáta určené ku klasifikácií. V učiacej fáze prebieha iba ukladanie vstupných už klasifikovaných dát, ale ich normalizácií. Vlastný model sa vytvára až v dobe klasifikácie. Algoritmus k-najbližších susedov je založený na porovnávaní vzdialenosti dvoch bodov, ak každá N-tica učiaca množiny predstavuje jeden bod v N-dimenzionálnom priestore. Pre výpočet vzdialenosti sa používa vzorec A.4.

$$\begin{aligned} \mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) &= \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}. \end{aligned} \tag{A.4}$$

Euklidovská vzdialenosť medzi bodmi p a q je vzdialenosťou časti úsečky spojujúcu $\overline{\mathbf{pq}}$. V kartezskej súradnicovej osi, ak $p = (p_1, p_2, ..., p_n$ a $q = (q_1, q_2, ..., q_n$ sú dva body v

Euklidoskom N-dimenzionálnom priestore, potom vzdialenosť (d), od p ku q, alebo naopak je popísaná pytagorovou formulou vo vzorci A.4. Euklidovská metrika je použiteľná iba v prípade že máme k dispozícií iba numerické atribúty. Pri iných ako numerických dáta je možné použiť vzorec pre Hammingovú vzdialenosť pre reťazce. Hammingovú vzdialenosť môžeme použiť aj pri normalizácií za predpokladu že zhodné hodnoty atribútov budú mať vzdialenosť 0 a odlišní vzdialenosť 1.

Klasifikácia spočíva vo výpočte vzdialeností skúmaného bodu od všetkých bodov v učiacej sa množine a následnom určení triedy. Trieda našeho bodu je tá, ktorá má najväčšie zastúpenie medzi jeho k-najbližšími susedmi[11].



Obr. A.6: Príklad klasifikácie ze pomocí algoritmu KNN

Na obrázku A.6 je možné vidieť klasifikáciu pre viaceré hodnoty k. Vyšetrovaným bodom je kruh v strede. Tento bod môže byť klasifikovaný ako trojuholník, alebo štvorec. Ak sa k rovná 3 (neprerušovaná kružnica), tak bude náš bod klasifikovaný ako trojuholník, pretože v priestore $k = 3$ sa nachádzajú dva trojuholníky a iba jeden štvorec. Ak by sme ale uvažovali o priestore, ktorý označuje $k = 5$ (prerušovaná kružnica), tak by náš bod bol klasifikovaný ako štvorec, lebo sa v tomto priestore nachádzajú tri štvorce a iba dva trojuholníky.

Algoritmus KNN nieje implementačné náročný a je vhodný predovšetkým ako referenčný model. Bohužiaľ je pomalý v rozhodovacej fáze, výpočetne náročný a závislý na použitej metrike a vhodne zvolenej konštante k. Pred použitím je vhodné dáta z učiacej množiny normalizovať, napríklad pomocou min-max normalizácie, ktorá prevedie vstupné atribúty na hodnoty v intervale $X =< 0.0, 1.0 >$. Následne je potrebné zvoliť konštantu k, tak aby dochádzalo ku čo najmenšej chybovosti.

A.2 Učenie bez učiteľa - Unsupervised learning

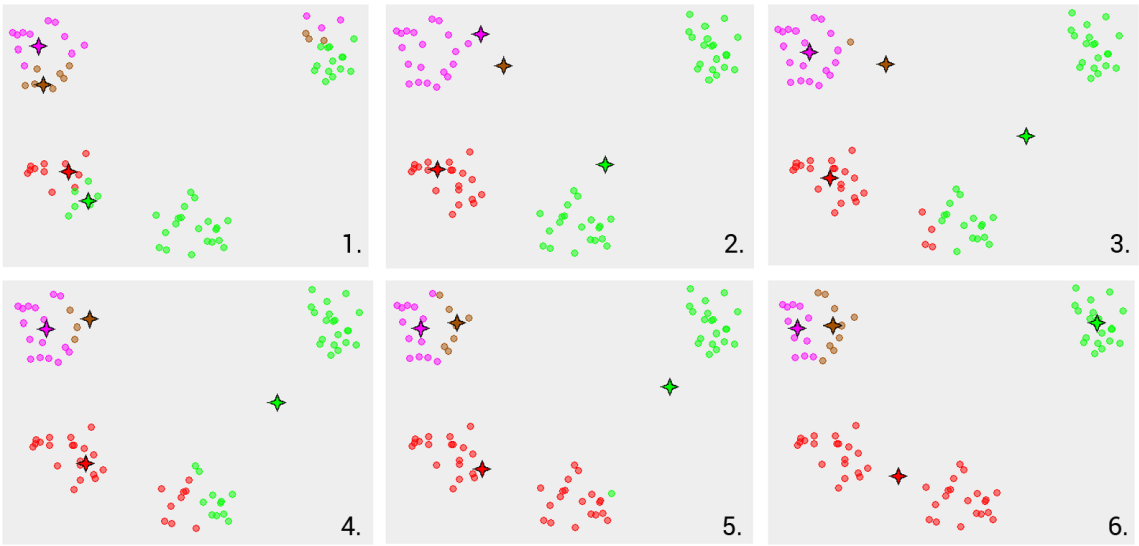
Učenie bez učiteľa nepotrebuje predom klasifikovanú množinu dát, učiacu množinu a prípadne nemusí poznať ani počet klasifikovaných tried[6]. Táto metóda sa využíva ku klasterizácií v prípadoch, kedy potrebujeme zoskupiť dáta podľa ich atribútov, keď sa snažíme vo vstupných dátach objaviť novú triedu alebo zistiť koľko tried a s akými vlastnosťami sa v dátach vyskytujú. Pre metódy učenia bez učiteľa je typické vytváranie zhlukov. Medzi najznámejších zástupcov tejto kategórie patria metódy K-mean a K-menoids.

A.2.1 K-means

Tento nehierarchický algoritmus slúži na triedenie n-dimenzionálnych dát do k zhlukov. Triedenie prebieha pomocou vybraných metrických funkcií, spôsobom priradenia bodov ku zhluku, ku ktorému stredu má najbližšie. Platí že počet stredov zhlukov je menší ako počet dát, teda bodov, ktoré budeme triediť. Metóda potrebuje vedieť počet zhlukov a dát, ktoré má triediť. Algoritmus náhodne vyberá zo vstupných dát daný počet bodov, ktoré budú tvoriť stredy zhlukov. Následne pre všetky ostatné body spočíta vzdialenosť a priradí ju ku zhluku, ku ktorému má tento body najbližšie. Následne vypočíta nový stred zhluku pomocou bodov, ktoré sa v zhluku nachádzajú a za pomoci novo získanej strednej hodnoty výpočet opakuje. Výpočet sa bude opakovať tak dlho pokiaľ bude dochádzať ku prenosom bodov medzi zhlukmi. To znamená že výpočet bude prebiehať pokiaľ hodnota nebude konvergovať[9][10].

$$E = \sum_{i=1}^{k} \sum_{\mathbf{p} \in C_i} |\mathbf{p} - \boldsymbol{m}_i|^2 \tag{A.5}$$

Vzorec A.5 popisuje kvadratickú chybu, ktorá sa využíva v tejto funkcií. Hodnota E vyjadruje súčet kvadratických chýb pre všetky body zo vstupných dát. Počítam teda pre každý objekt p v zhluku C_i jeho vzdialenosť od stredu m. Následne ich sčítame. Cieľom je vytvoriť práve k zhlukov, ktoré budú obsahovať minimálne vzdialenosti bodov od ich stredov a naopak, kde budú vzdialenosti medzi zhlukmi a ich bodmi najväčšie.



Obr. A.7: K-Means konvergujúca na lokálne minimum

Na obrázku A.7 je vyobrazený príklad zhlukovania pomocou K-means algoritmu. Hviezdami sú označené stredy zhlukov. Na obrázku metóda konverguje po piatich iteráciách na svoje minimum. Metód K-means pracuje dobre nad dátami, ktoré sú sú podľa podobností zoskupené pri sebe. Nepracuje dobre nad dátami, ktoré majú príliš veľké odlišnosti, preto je citlivá na chybné a poškodené dáta.

A.2.2 K-menoids

Metóda K-menoids je podobná metóde K-means. Rozdielom týchto metód je fakt že K-menoids používa ako stred zhluku, niektorý z bodov vnútri zhluku. Celý algoritmus je následný totožný z K-means. Pri aktualizácií stredu zhluku môže dôjsť ku určeniu nového bodu, ktorý bude predstavovať stred. Ak je zmenená stred zhluku, musí dôjsť ku premiestneniu bodov patriaceho do zhluku. Vďaka tomu že K-menoids používajú ako stred zhluku samotné body, tak niesu citlivé voči poškodením dátam, ale naopak práve takéto dáta budú tvoriť stredy zhlukov. K-menoids je teda robustnejšou nadstavbou nad K-means[10].

A.3 Kombinované učenie - Semi-supervised learning

Semi-supervised learning, alebo kombinované učenie v sebe zahŕňa špecifické vlastnosti učenia s učiteľom a učenia bez učiteľa. Metódy pracujú na princípe kombinovaného učenia a nespoliehajú sa vo svojej učiacej fáze na veľké množstvo predom klasifikovaných dát. Sú veľmi vhodné tam, kde je nedostatok pred pripravených dát, alebo je veľmi ťažké takéto dáta získať. Medzi najpoužívanejšie partia Self-training a Co-trainig.

A.3.1 Self-training

Tento algoritmus si pomocou klasifikovaných dát vytvori klasifikátor, ktorý následne používa pre identifikáciu neklasifikovaných vstupných dát. Následne sú najdôveryhodnejšie neklasifikované dáta pridané ku učiacej množine a znovu dôjde ku vytvorení nového modelu a celý proces sa opakuje. Klasifikátor tu využíva dáta, ktoré sa sám naučil a klasifikoval pre opätovné učenie.

Algoritmus sa dá popísať aj formálne. Nech L je množina klasifikovaných dát, U množina neklasifikovaných dát a h je klasifikátorom. Opakujeme proces:

- Nauč klasifikátor h pomocou dát L,
- Klasifikuj dáta U pomocou h,
- Nájdi podmnožinu $U\prime$ z množiny U s najdôveryhodnejšími hodnotami,
- $L + U\prime \rightarrow L$,
- $U - U\prime \rightarrow U$.

A.3.2 Co-training

Pri tomto algoritme je učiaca množina rozdelení na niekoľko vzájomne nezávislých podmnožín, pre každú podmnožinu je z jej klasifikovaných dát vytvorený model, ktorý je použitý na následnú identifikáciu neklasifikovaných dát z podmnožiny. Klasifikátory si vzájomne vymieňajú najviac dôveryhodné klasifikované dáta spoločne z ich predikovanou triedou a

dochádza ku opätovnému preučeniu klasifikátorov. V podstate sa množiny rozdelia na niekoľko častí a každá množina je klasifikovaná pomocou klasifikátoru a tento klasifikovaný výstup sa predá inému klasifikátoru na preučenie. Co-training je odolnejší voči chybám ako Self-training. Pri učení a klasifikácií dát neobsahuje informácie o ich triede.

Variácie Co-trainingu:

- Goldman a Zhou (2000) používal dve učiace sa množiny rôznych typov, ktoré ale mali na vstupe množinu všetkých dát.
- Zhou a Li (2005) používal tri plastifikátory. Ak sa dva zhodnú tak predajú dáta tretiemu na učenie.

Príloha B

Kompletné informácie o testovaní

Tabuľka B.1: Počet konverzácií zo štítkom

Štítok aplikácie	L7 Konverzácie
443_TCP_Https-iexploreexe	10998
5355_UDP	10980
443_TCP_Https-system	9309
53_UDP_Dns-	7888
80_TCP_HTTP-	6173
443_TCP_Https	3307
27019_UDP	3026
443_TCP_Https-skypeexe	2899
1900_UDP	2862
80_TCP_HTTP-iexploreexe	2383
443_TCP_Https-itunesexe	2178
443_TCP_Https-spotifyexe	1640
80_TCP_HTTP-spotifyexe	1590
2869_TCP	656
27018_UDP	571
27017_UDP	557
80_TCP_HTTP-steamexe	529
3128_TCP-steamwebhelperexe	489
27020_UDP	435
443_TCP_Https-skypebrowserhostexe	385
2869_TCP-system	325
27021_UDP	204
443_TCP_Https-svchostexe	171
3478_UDP	156
43265_UDP	139
80_TCP_HTTP-itunesexe	128
5351_UDP	98
138_UDP	87
137_UDP	84
67_UDP_DhcPs	79
443_UDP_Https	74
139_TCP-system	66
443_TCP_Https-steamexe	61
3702_UDP	56
4070_TCP-spotifyexe	55
443_TCP_Https-steamwebhelperexe	54
40029_UDP	53

Tabuľka B.2: ML nová implementácia

Štítok aplikácie	F- Miera 0.7	F- Miera 0.9	L7 Konverzácie
137_udp	0.080	0.000	84
138_udp	1.000	1.000	87
139_tcp-system	0.000	0.000	66
1900_udp	0.992	0.980	2862
27017_udp	0.000	0.105	557
27018_udp	0.000	0.961	571
27019_udp	0.998	1.000	3026
27020_udp	0.000	0.000	435
27021_udp	0.000	0.000	204
2869_tcp	0.992	1.000	656
2869_tcp-system	0.995	1.000	325
3128_tcp-steamwebhelperexe	0.951	0.965	489
3478_udp	1.000	1.000	156
3702_udp	0.000	0.000	56
40029_udp	0.500	0.910	53
4070_tcp-spotifyexe	0.000	0.000	55
43265_udp	0.667	0.678	139
443_tcp_https	0.689	0.894	3307
443_tcp_https-iexploreexe	0.851	0.998	10998
443_tcp_https-itunesexe	0.879	1.000	2178
443_tcp_https-skypebrowserhostexe	0.000	0.000	385
443_tcp_https-skypeexe	0.849	1.000	2899
443_tcp_https-spotifyexe	0.343	1.000	1640
443_tcp_https-steamexe	0.000	0.986	61
443_tcp_https-steamwebhelperexe	0.000	0.000	54
443_tcp_https-svchostexe	0.000	0.000	171
443_tcp_https-system	0.445	0.998	9309
443_udp_https	0.000	0.000	74
53_udp_dns	0.000	0.994	7888
5351_udp	0.991	0.000	98
5355_udp	0.990	1.000	10980
67_udp_dhcps	0.950	1.000	79
80_tcp_http	0.861	1.000	6173
80_tcp_http-iexploreexe	0.302	0.979	2383
80_tcp_http-itunesexe	0.000	0.000	128
80_tcp_http-spotifyexe	0.931	1.000	1590
80_tcp_http-steamexe	0.451	1.000	529

Tabuľka B.3: EPI nová implementácia

Štítok aplikácie	F-miera 0.7	F-miera 0.9	L7 Konverzácie
137_udp	0.074	0.000	84
138_udp	0.426	0.720	87
139_tcp-system	0.947	0.923	66
1900_udp	0.000	0.000	2862
27017_udp	0.000	0.000	557
27018_udp	0.033	0.032	571
27019_udp	0.000	0.000	3026
27020_udp	0.000	0.000	435
27021_udp	0.046	0.000	204
2869_tcp	0.000	0.000	656
2869_tcp-system	0.929	1.000	325
3128_tcp-steamwebhelperexe	0.900	0.819	489
3478_udp	0.638	0.609	156
3702_udp	0.791	0.706	56
40029_udp	0.286	0.308	53
4070_tcp-spotifyexe	0.882	1.000	55
43265_udp	0.689	0.643	139
443_tcp_https	0.360	0.365	3307
443_tcp_https-iexploreexe	0.582	0.543	10998
443_tcp_https-itunesexe	0.696	0.653	2178
443_tcp_https-skypebrowserhostexe	0.172	0.182	385
443_tcp_https-skypeexe	0.482	0.460	2899
443_tcp_https-spotifyexe	0.299	0.233	1640
443_tcp_https-steamexe	0.000	0.000	61
443_tcp_https-steamwebhelperexe	0.000	0.000	54
443_tcp_https-svchostexe	0.239	0.091	171
443_tcp_https-system	0.621	0.596	9309
443_udp_https	0.000	0.000	74
53_udp_dns	0.694	0.696	7888
5351_udp	0.811	0.769	98
5355_udp	0.977	0.978	10980
67_udp_dhcps	1.000	1.000	79
80_tcp_http	0.685	0.679	6173
80_tcp_http-iexploreexe	0.359	0.297	2383
80_tcp_http-itunesexe	0.000	0.000	128
80_tcp_http-spotifyexe	0.848	0.838	1590
80_tcp_http-steamexe	0.599	0.548	529

Tabuľka B.4: ML pôvodná implementácia

Štítok aplikácie	F- Miera 0.7	F- Miera 0.9	L7 Konverzácie
137_udp	0.207	1.000	84
138_udp	1.000	1.000	87
139_tcp-system	1.000	1.000	66
1900_udp	0.000	0.995	2862
27017_udp	0.351	0.000	557
27018_udp	0.011	1.000	571
27019_udp	0.833	1.000	3026
27020_udp	0.039	1.000	435
27021_udp	0.000	0.000	204
2869_tcp	0.485	1.000	656
2869_tcp-system	0.000	1.000	325
3128_tcp-steamwebhelperexe	0.749	1.000	489
3478_udp	0.978	1.000	156
3702_udp	0.850	1.000	56
40029_udp	0.800	0.667	53
4070_tcp-spotifyexe	0.903	1.000	55
43265_udp	0.635	0.740	139
443_tcp_https	0.046	0.000	3307
443_tcp_https-iexploreexe	0.488	1.000	10998
443_tcp_https-itunesexe	0.706	1.000	2178
443_tcp_https-skypebrowserhostexe	0.183	0.000	385
443_tcp_https-skypeexe	0.474	1.000	2899
443_tcp_https-spotifyexe	0.000	1.000	1640
443_tcp_https-steamexe	0.000	0.000	61
443_tcp_https-steamwebhelperexe	0.000	0.000	54
443_tcp_https-svchostexe	0.492	0.000	171
443_tcp_https-system	0.119	0.000	9309
443_udp_https	0.894	0.000	74
53_udp_dns	0.988	0.998	7888
5351_udp	0.104	1.000	98
5355_udp	0.967	1.000	10980
67_udp_dhcps	0.957	1.000	79
80_tcp_http	0.548	1.000	6173
80_tcp_http-iexploreexe	0.315	1.000	2383
80_tcp_http-itunesexe	0.029	0.000	128
80_tcp_http-spotifyexe	0.882	0.992	1590
80_tcp_http-steamexe	0.552	0.666	529

Tabuľka B.5: EPI pôvodná implementácia

Štítok aplikácie	F-miera 0.7	F-miera 0.9	L7 Konverzácie
443_tcp_https-iexploreexe	0.556	0.568	10998
5355_udp	0.963	0.962	10980
443_tcp_https-system	0.093	0.077	9309
53_udp_dns	0.989	0.991	7888
80_tcp_http	0.601	0.640	6173
443_tcp_https	0.058	0.051	3307
27019_udp	0.840	0.854	3026
443_tcp_https-skypeexe	0.433	0.489	2899
1900_udp	0.000	0.000	2862
80_tcp_http-iexploreexe	0.401	0.444	2383
443_tcp_https-itunesexe	0.676	0.689	2178
443_tcp_https-spotifyexe	0.000	0.000	1640
80_tcp_http-spotifyexe	0.895	0.919	1590
2869_tcp	0.473	0.517	656
27018_udp	0.012	0.000	571
27017_udp	0.000	0.000	557
80_tcp_http-steamexe	0.633	0.640	529
3128_tcp-steamwebhelperexe	0.864	0.864	489
27020_udp	0.299	0.275	435
443_tcp_https-skypebrowserhostexe	0.169	0.167	385
2869_tcp-system	0.234	0.000	325
27021_udp	0.000	0.000	204
443_tcp_https-svchostexe	0.140	0.100	171
3478_udp	1.000	1.000	156
43265_udp	0.779	0.615	139
80_tcp_http-itunesexe	0.000	0.000	128
5351_udp	0.116	0.108	98
138_udp	1.000	1.000	87
137_udp	0.074	0.200	84
67_udp_dhcps	0.909	1.000	79
443_udp_https	0.773	0.625	74
139_tcp-system	1.000	1.000	66
443_tcp_https-steamexe	0.000	0.000	61
3702_udp	1.000	1.000	56
4070_tcp-spotifyexe	0.800	1.000	55
443_tcp_https-steamwebhelperexe	0.000	0.000	54
40029_udp	0.718	0.615	53

Príloha C

Obsah CD

Obsah jednotlivých zložiek na priloženom CD:

- *src* - Zdrojové súbory aplikácie
- *doc* - Zdrojové súbory bakalárnej práce
- *xchomo00.pdf* - Bakalárna práca vo formáte PDF
- *README* - Popis súborov aplikácie